Philippe DAX



PARANINFO SA

CP/M
Guía de utilización

MP/M
CP/M 86
WORDSTAR
BASIC

Philippe DAX

CP/M
Guía de utilización

MP/M
CP/M 86
WORDSTAR
BASIC

1986



MADRID

Traducido por:
FELISA MATEO GARCIA

Esta obra es la traducción del libro francés de "Philippe Dax" titulado CP/M ET SA FAMILLE Guide d'utilisation, publicado por Editions EYROLLES París (Francia)

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1430-6

Depósito Legal: M-35047-1985



Magallanes, 25 - 28015 MADRID (3-3473)

ALCO, artes gráficas. Jaspe 34. 28026 Madrid

Indice

Prefacio

El sistema CP/M, muy popular en Estados Unidos, se ha convertido, desde los primeros años 80, en un verdadero "estándar" para los micro-ordenadores basados en microprocesadores de ocho bits. Su influencia se ejerce especialmente en la micro-informática individual y doméstica, aunque también está presente en la informática profesional y en la gestión de las PME.

Este éxito se lo debe el CP/M a sí mismo, así como a la gran variedad de productos software (paquetes de programas) que se han desarrollado alrededor de él, en el momento adecuado. Esta amplia gama de productos, cuyas áreas de aplicación se encuentran muy extendidas y diversificadas, cubren: los lenguajes de programación, el tratamiento de textos, el desarrollo del Software, los programas de gestión, las comunicaciones entre ordenadores y las redes.

Este "maremoto CP/M" que se ha desencadenado al otro lado del Atlántico, ha empezado ya a hacer mella en Europa y muy especialmente en Francia.

Es por esta razón por lo que me ha parecido conveniente dar a conocer este producto, sus extensiones y su entorno de programación más detalladamente para todos aquellos que deseen estudiarlo, experimentar y practicar con él.

Este libro se divide en cuatro partes: el sistema mono-usuario CP/M, el sistema multi-usuario MP/M, las extensiones actuales nacidas del CP/M, MP/M y de la llegada de los microprocesadores de 16 bits, y por último, una visión general de los productos desarrollados alrededor de lo que se podría llamar "la familia CP/M".

Para intentar responder a las necesidades y exigencias de cada lector, se hace, al principio de las tres primeras partes, una aproximación más general, relativa a la estructura funcional de los sistemas para destacar más el aspecto descriptivo y de "guía de utilización".

Aunque este libro no tenga la pretensión de ser exhaustivo en lo que respecta al conjunto de mandatos estándares del CP/M y de los productos que existen alrededor de él, pienso que el lector, ya sea amateur, estudiante o profesional, encontrará en él los elementos que precise.

1

El sistema CP/M

1.1. INTRODUCCION

El CP/M (Control Program for Microcomputer) es un sistema operativo mono-usuario difundido universalmente en el mundo de la microinformática.

El sistema operativo CP/M está destinado para los micro-ordenadores basados en microprocesadores de palabras de ocho bits tipo 8080 de Intel y Z80 de Zilog. El CP/M equipa a otros microprocesadores e incluso a algunos de "16" bits. Más adelante estudiaremos estos casos particulares, así como las evoluciones actuales y futuras de los productos CP/M y de su familia. Por el momento, nos limitaremos al estudio del sistema CP/M clásico, es decir, de las versiones comercializadas (releases) con los identificativos: CP/M 1.4, CP/M 2.0 y CP/M 2.2.

Antes de abordar en detalle la descripción, el funcionamiento y las facilidades ofrecidas por el sistema CP/M, retrocedamos un poco y hagamos un comentario histórico con el fin de situar mejor al CP/M.

1.1.1. Reseña histórica del CP/M

El CP/M fue concebido en 1973 por Gary Kildall, en una época en la que la micro-informática, que todavía estaba en estado embrionario, estaba reservada para algunos adelantados. La idea de G. Kildall fue concebir un sistema operativo que respondiese a los siguientes criterios:

- Adaptabilidad a cualquier micro-ordenador construido sobre microprocesadores del tipo 8080, 8085 ó Z80.

– Posibilidad de almacenamiento y archivo de los datos y los programas de los usuarios en las memorias auxiliares con bajo costo (disquetes o discos-floppy).
– Provisión de una interfase de programación, completa y estándar, con todas las funciones necesarias para un programador.

Hay una frase de G. Kildall que expresa bastante bien lo que ha querido hacer con el CP/M: "CP/M es el análogo software del bus S-100".

Kildall funda la Digital Research en 1976, fecha a partir de la cual se emprende el desarrollo y la comercialización del CP/M.

1.1.2. Evolución actual del CP/M

Actualmente el sistema CP/M ha tomado una dimensión tal que, de hecho, se ha convertido en un estándar incuestionable (como sucede con el bus S-100 en el aspecto físico).

Se calcula en más de trescientos mil el número de usuarios del sistema CP/M, sin contar las no menos de cuatrocientas a quinientas sociedades OEM (Original Equipment Manufacturer), varias decenas están en Francia, que han desarrollado unos micro-ordenadores que funcionan bajo CP/M. Entre los principales constructores que han adoptado el CP/M, se encuentran: Altos, Digital Equipment, Hewlett Packard, IBM, ICL, Intel, ITT, Mostek, National Semiconductor, REE, Sharp, Xerox, Zenith, Zilog...

No obstante, este sistema, aunque universalmente conocido, también tiene sus detractores. En ciertos puntos se le pueden hacer algunas críticas, tanto a la filosofía del sistema propiamente dicho, como a la realización concreta de algunos de sus elementos. Pero el éxito lo lleva más allá de sus insuficiencias. El mismo G. Kildall lo dice: "Conozco verdaderos especialistas en sistemas que, después de haber analizado el CP/M con lupa, no han quedado impresionados". Hablando con propiedad, el CP/M no es un modelo en su género, sino un sistema que "se ajusta", gusta y se beneficia de una amplia biblioteca de programas que van desde los lenguajes de programación a los paquetes de gestión, pasando por herramientas de software muy diversificadas como el tratamiento de textos, las comunicaciones entre ordenadores, las bases de datos, etc...

El porvenir del CP/M, vista su facilidad de adaptación, tiene todas las posibilidades de ser floreciente, ya que se encuentra en la bien

definida "almena" de la informática individual, ya sea doméstica o profesional.

En contrapartida, el CP/M corre el peligro de entrar en competencia con la aparición y el auge de los sistemas multi-usuarios de uso general (general purpose), tipo UNIX. Estos sistemas, destinados a gestionar varios usuarios simultáneamente, equipan a los micro-ordenadores más potentes, basados en microprocesadores de 16 y 32 bits.

De cara a esta competencia inevitable, Digital Research ha encontrado la respuesta con unos productos procedentes del CP/M y compatibles con él como el MP/M, MP/M-II, CP/M-86, MP/M-86, CP/NET, MP/NET, etc..., productos que analizaremos más adelante en este libro.

	Mono-usuario mono-tarea	Mono-usuario multi-tarea	Multi-usuario multi-tarea	Red
8 bits	CP/M	MP/M	MP/M MP/M-II	CP/NET MP/NET
16 bits	CP/M-86	MP/M-86 CCP/M-86	MP/M-86	CP/NET-86 MP/NET-86

Fig. 1.— Distribución de la familia CP/M.

1.2. LA CONFIGURACION FISICA REQUERIDA POR EL CP/M

1.2.1. Microprocesadores

La concepción modular del CP/M, le permite adaptarse a aquellas máquinas realizadas en base a microprocesadores tipo 8080, 8085 y Z80, que poseen un código operativo común (juego de instrucciones del código máquina).

El conjunto de los modelos base del CP/M (CCP, BDOS, BIOS), cuyo análisis está detallado más adelante, está escrito en el ensamblador 8080 o en el lenguaje estructurado PL/M; el resultado que se produce es el lenguaje máquina 8080 (subconjunto de los lenguajes del 8085 y Z80). El Z80, más potente a nivel de código operativo que el del

8080, estará pues infra-utilizado. Pero nada impide al usuario utilizar las instrucciones del Z80 si el micro-ordenador viene equipado con este microprocesador. Al sistema CP/M se le llama "machine dependant", es decir, dependiente de la CPU, debido al lenguaje máquina. El CP/M, por lo tanto, no es transportable sobre cualquier máquina como, en teoría, lo son los sistemas UCSD y UNIX escritos en un lenguaje evolucionado: Pascal para UCSD y C para UNIX.

Para el CP/M, no se puede hablar de transportabilidad de los productos más que entre máquinas que evolucionan en un contexto CP/M. Sólo en este caso se pueden transportar los objetos o los binarios de una máquina a otra.

La profusión de micro-ordenadores que funcionan bajo CP/M, demuestra que el CP/M es un sistema fácilmente implantable a pesar del hándicap que supone la imposición de las CPU's. De hecho, el CP/M tiene la ventaja de ser independiente de la configuración física que rodea al microprocesador. Es este entorno el que vamos a describir.

1.2.2. La memoria central

El sistema propiamente dicho sólo ocupa 6,5 Kilo-octetos de memoria central RAM (Random Access Memory). Pero hay un cierto número de productos estándares (editor de texto ED, debugger DDT, ensamblador ASM, etc.) que funcionan en el contexto del CP/M y se entregan con él, que necesitan un mínimo de 16 Kilo-octetos. El tamaño de la memoria puede llegar hasta 64 Kilo-octetos en la configuración máxima.

1.2.3. La memoria secundaria

El CP/M necesita además una memoria llamada secundaria que no es más que un soporte magnético directamente direccionable, es decir, uno o varios discos. Normalmente los sistemas de micro-informática están equipados con disquetes o discos flexibles (floppys).

1.2.3.1. Los disquetes o discos flexibles

Existen dos tipos de disquetes reconocibles por sus dimensiones: los disquetes de 5 1/4 pulgadas de diámetro y los disquetes de 8 pulgadas estos últimos más utilizados en el campo profesional. La capacidad de los disquetes puede variar según su diámetro.

Otra característica importante de los disquetes es la densidad. Generalmente distinguimos dos: simple densidad (26 sectores por pista) y doble densidad (48 ó 52 sectores por pista). Además en algunos disquetes se pueden utilizar las dos caras magnéticas con lo que se incrementa otro tanto la capacidad total de almacenamiento.

Los disquetes de 8 pulgadas son compatibles con el formato IBM 3740 para mejorar la portabilidad de los productos. En efecto, el formateo de los disquetes CP/M es el mismo que los de IBM pero las informaciones no son compatibles (organización de los directorios y de los ficheros, datos en EBCDIC, ...). No obstante, existen unas utilidades que permiten convertir su contenido y pasar de un formato a otro.

1.2.3.2. Los discos duros

Los discos "duros" o rígidos también pueden sustituir a los disquetes proporcionando una gran capacidad que puede ir de 5 a 40 Megaoctetos (tipo Winchester o Cynthia). Al igual que para los disquetes, existen varios formatos de discos duros: 5 1/4 pulgadas, 8 pulgadas, 13 pulgadas. El disco duro se encuentra encerrado en un recinto hermético para evitar todo contacto con el polvo. Aunque los brazos de acceso a las pistas sean móviles, el disco está fijo e integrado en la máquina. También existen discos duros con una parte móvil y otra fija, permitiendo así cambiar el "pack" (paquete) de discos, en caso necesario.

1.2.4. La consola, puesto de trabajo interactivo

Dado que el sistema CP/M es, por definición, un sistema mono-puesto, impone la presencia de un terminal conversacional. Este terminal, llamado también "consola" o "pantalla" en la terminología informática, suele ser normalmente un terminal tipo "CRT" (Cathode Ray Tube), es decir, un conjunto formado por una pantalla y un teclado alfanumérico.

1.2.5. Los periféricos anexos

Dependiendo de la configuración física adoptada para el micro-ordenador, éste puede disponer de "puertas" (ports) de entrada-salida específicas o auxiliares. Concretamente para la salida a impresora, podemos tenerla sobre un "port" serie o sobre un "port" paralelo (8 bits).

Otros "ports" series o interfases RS232 C (V24 del CCITT) pueden formar parte del conjunto, ya sea en forma estándar o como opción a implantar.

1.3. DESCRIPCION FUNCIONAL DEL CP/M

En todas las máquinas en las que se ha implantado el CP/M existe un "cold start loader", sistema mínimo en "PROM" que se encarga de realizar el "bootstrap" (arranque o inicialización del sistema). Esta operación inicializa el acoplador de disquetes y carga en la memoria "RAM" los módulos del sistema operativo (CCP, BDOS, BIOS).

El sistema CP/M está pues constituido por tres módulos funcionales:

- CCP: Console Command Processor: procesador de mandatos de consola;
- BDOS: Basic Disk Operating System: sistema de la gestión básica de los ficheros en disco;
- BIOS: Basic Input Output System: sistema de entradas-salidas básicas.

Podemos considerar que el CCP es la interfase hombre/máquina, el BDOS el conjunto formado por los módulos de la gestión lógica de la memoria secundaria y el BIOS el núcleo físico del CP/M.

El conjunto conjugado del CCP y el BDOS constituye el núcleo lógico del sistema CP/M suministrado por Digital Research, independiente del entorno externo y por lo tanto transportable sobre cualquier máquina basada en el 8080 o en el Z80.

Por el contrario, el BIOS, que contiene los programas que se comunican directamente con las unidades físicas, está escrito por el constructor del micro-ordenador para adaptarlo al entorno físico.

1.3.1. El CCP

El CCP se divide en dos partes funcionales:

- El intérprete de mandatos y
- Los mandatos integrados.

1.3.1.1. El intérprete de mandatos

El CCP es, esencialmente, un intérprete de mandatos. En efecto, se encarga de leer las órdenes tecleadas por el usuario desde la consola y las analiza sintácticamente antes de proceder a la ejecución del programa. Para hacer ésto, el CCP utiliza las funciones realizadas por los otros módulos del sistema: el BIOS y el BDOS. Precisamente el diálogo con la consola del usuario lo realiza el CCP conjuntamente con el BIOS filtrándose algunos caracteres especiales para poder ofrecer al usuario algunas funciones de edición: borrado de un carácter o de una línea, fin de mensaje, fin de línea...

Una vez que el mandato se ha aceptado, desde el punto de vista sintáctico, se carga el fichero tipo COM en la zona TPA reservada para el programa del usuario. Desde el momento en que el control se pasa al programa del usuario y durante su fase de ejecución, el CCP queda inoperativo. Algunos programas muy voluminosos aprovechan el hecho de que el CCP quede inactivo para desbordar la zona de la TPA y ocupar una parte del área asignada al CCP, contigua a la TPA. En este caso, el CCP se destruye por "solapamiento" aunque de una manera temporal. En efecto, la terminación de la ejecución de un programa, ya sea normal, ya sea forzada (<ctrl-C> desde el teclado), provoca una reinicialización del sistema llamada "arranque en caliente" ("warm start"). Esta reinicialización vuelve a traer el CCP desde el disco a memoria, transmitiéndose entonces al usuario una nueva invitación para que teclee un mandato.

La invitación para el tecleo de un mandato viene señalada por la aparición en el terminal de un "prefijo" o "prompt" de la siguiente forma:

A>   donde "A" representa el nombre del disco actual

1.3.1.2. Convenios de utilización de los mandatos

En este libro, en lo que respecta a las ilustraciones de los ejemplos, para diferenciar los mensajes emitidos por la máquina ("prompt", respuestas) de lo que el usuario escribe desde el teclado (mandatos), adoptaremos los siguientes convenios:

- Los mandatos introducidos desde el teclado aparecerán subrayados.
- El retorno de carro, identificado por <CR>, exigido para terminar un mandato, no aparecerá excepto en el caso en que sea precisa una aclaración.

– Los caracteres especiales que se obtienen al pulsar simultáneamente la tecla "CONTROL" y el carácter especificado vendrán representados por los símbolos "ctrl" o "↑" seguidos del carácter en cuestión, ejemplo: <ctrl-C> o ↑C.

1.3.1.3. Sintaxis de un mandato

Un mandato, ya sea del sistema o del usuario, se presenta como una cadena de caracteres, opcionalmente separados por espacios (blancos), que termina obligatoriamente con un retorno de carro (tecla "Return" del teclado). La cadena de caracteres que sigue inmediatamente al "prompt" representa el nombre propiamente dicho del mandato y las cadenas siguientes los argumentos o los parámetros asociados.

Existen dos categorías de mandatos bajo CP/M: los mandatos residentes integrados en el sistema en el módulo CCP y los mandatos no residentes grabados en disco en forma de ficheros estándares con el tipo COM que les otorga la condición de ficheros ejecutables.

Los argumentos pueden representar nombres de ficheros (ver la sección "referencia a un fichero"), palabras reservadas o valores numéricos.

En el caso de los mandatos no residentes, se puede espeficicar el nombre del disco sobre el cual está situado el fichero ejecutable asociado, sin precisar su tipo COM que está implícito.

El siguiente ejemplo:

```
A>B: STAT        A:FICHERO.ASM
```

indica que el usuario ha ido a buscar el fichero "STAT. COM" sobre el disco "B" para conocer los avisos concernientes al "FICHERO.ASM" del disco "A".

En la entrada de datos, el CP/M acepta indistintamente las letras mayúsculas o minúsculas, convirtiéndolas sistemáticamente en mayúsculas.

En lo que respecta al funcionamiento interno, el CP/M ejecuta directamente el mandato si éste es residente, si no lo es, busca en el disco actual el nombre asociado del fichero ejecutable de tipo COM cargando este fichero en el área TPA de la memoria; a continuación, en caso necesario, se encarga de abrir el primer fichero puesto como argumento.

1.3.1.4. Selección de un disco

Cada disco que trabaje bajo CP/M contiene su propio directorio (directory). Este directorio da la información del conjunto de ficheros existentes en el disco.

Si el sistema dispone de varios soportes de discos (discos flexibles o disquetes y discos duros), el usuario puede seleccionar fácilmente el soporte sobre el que desea trabajar. Basta con teclear el nombre del disco inmediatamente a continuación del "mensaje de diálogo" ("prompt") mediante una de las letras A, B, ..., P seguida del carácter ":", para los discos numerados 0,1, ..., 15.

Ejemplo de cambio de utilización de discos:

A > B:	el usuario selecciona el disco "B"
B >	nuevo "prompt" emitido por el CCP
B > A:	el usuario vuelve a seleccionar el disco "A"
A >	

1.3.1.5. Concepto de usuario

En la versión 1.4 del CP/M no existe el concepto de propietario y cualquier usuario puede acceder a todos los ficheros del/de los discos conectados en ese momento al sistema. El concepto de propiedad de la información está ligado al de la propiedad física del soporte.

En la versión CP/M 2.2 se ha incorporado una mejora para permitir que varios usuarios tengan cada uno su propio directorio en el mismo soporte, en la medida en que dichos usuarios comparten, por turnos, el tiempo de proceso del micro-ordenador y el espacio en disco.

El espacio en disco está, en efecto, segmentado en varias particiones lógicas llamadas "áreas de usuarios". Esta división del disco en áreas es completamente transparente para el usuario, ya que la segmentación no se realiza físicamente sobre el soporte, sino lógicamente a nivel del directorio general. El primer octeto de cada entrada queda entonces asociado al número de usuario (0 por defecto). Un usuario que se conecta puede entonces elegir su partición mediante el mandato:

A > USER n

donde n representa un número de área de usuario que puede variar entre 0 y 15.

La protección de la información no es efectiva, porque a este mandato puede acceder cualquier usuario, y el concepto de "palabra clave" no existe. Se puede contemplar esto como una deficiencia, pero hay que observar que el CP/M está concebido para ser un sistema operativo mono-usuario y que varios usuarios no pueden compartir la máquina simultáneamente.

Zona usuario 0
Zona usuario 2
Zona usuario N

Fig. 2.— Representación lógica de las áreas de usuarios.

1.3.1.6. Referencia a un fichero

Se accede a los ficheros gestionados por el BDOS, por medio de un identificador, cuyo formato general comprende tres campos de tipo cadena de caracteres, de la siguiente forma:

disco:fichero.tipo   Ejemplo: A:EJEMPLO.CPM

Este identificador representa una referencia de fichero que puede ser explícita o ambigua.

Una referencia explícita identifica a un único y determinado fichero, mientras que una referencia ambigua puede satisfacer a varios ficheros diferentes.

El primer campo representa el nombre del disco sobre el que desea posicionarse el usuario. Está simbolizado por una sola letra (de la "A" a la "P"); la letra "A" está asociada al disco 0 y soporta el sistema CP/M, la letra "B" al disco 1, ..., y la letra "P" al disco 15.

No es necesario indicar este identificador siempre y cuando el fichero se encuentre en el disco actual. Por lo tanto, la referencia a un fichero puede ser del tipo:

fichero.tipo   Ejemplo: FILE.ASC

Los otros dos campos representan el nombre principal del fichero y su tipo o extensión. El campo "tipo" es opcional, pero se utiliza generalmente para describir lo que contiene el fichero, y a este respecto,

puede ser muy útil. Por ejemplo ASM podría designar un programa escrito en ensamblador, OBJ un módulo objeto, COM un mandato ejecutable, etc...

Los dos campos del nombre están separados por un "." como vemos a continuación:

pppppppp.ttt — Ejemplo: NOMFICH.TEX

donde "pppppppp" representa el nombre principal (de uno a ocho caracteres), y "ttt" el tipo del fichero (de uno a tres caracteres). Como ya se ha mencionado, también se acepta el nombre:

pppppppp — Ejemplo:PROGRAMA

y su tipo por defecto es equivalente a tres blancos consecutivos.

Un nombre de fichero explícito no debe contener los caracteres especiales siguientes:

. , ; = ? * < > [ ]

estando permitidos el resto de caracteres alfanuméricos y los otros caracteres especiales.

+++---	es correcto	AZE;QWE	es incorrecto
!A"Z#E$R	es correcto	A.Z,E;R=	es incorrecto
(azerty)	es correcto	<qwerty>	es incorrecto

La referencia ambigua (o global) puede identificar a uno o a varios ficheros y se utiliza generalmente para buscar un fichero en un directorio hasta que se produce una coincidencia con una entrada del directorio. El formato de la referencia ambigua es similar al de la referencia explícita, con la única diferencia de que los caracteres "?" y "*" tienen unas funciones definidas que vamos a describir.

El carácter "?" permite la sustitución por cualquier otro carácter permitido en un nombre de fichero, en la posición indicada por este carácter. Así la referencia ambigua siguiente:

N?M.T??

satisface a los siguientes nombres de fichero (si existen):

NOM.TEX	? ⟶ O	y ?? ⟶ EX
NUM.TYP	? ⟶ U	y ?? ⟶ YP
NAM.TTY	? ⟶ A	y ?? ⟶ TY

pero no satisface a los siguientes nombres de fichero:

RAM.TPA	R falso
NAME.TTY	E falso
NAM.HEX	H falso

En cuanto al carácter "*", tiene el mismo significado que "?", pero en lugar de sustituir un único carácter en una posición dada, puede sustituir todo un campo, de forma que:

.

es equivalente a la referencia ambigua al fichero:

????????.???

es decir, a todos los ficheros presentes en el disco seleccionado. Del mismo modo:

pppppppp.*	es equivalente a:	pppppppp.???
*.ttt	es equivalente a:	????????.ttt

También es posible la utilización simultánea de "?" y "*", por ejemplo:

NOM???.*

1.3.1.7. Tipos predefinidos

Existe un cierto número de tipos predefinidos bajo CP/M que permiten especificar lo que representa el contenido del fichero: fuente simbólico, binario absoluto o reubicable, datos numéricos, información temporal, etc...

La siguiente lista muestra, aunque no en forma exhaustiva, los principales estándares de ficheros explotables bajo CP/M.

COM:	mandato ejecutable	HEX:	código máquina formato Intel
ASM:	fuente ensamblador 8080	PRN:	fichero imagen impresora
BAK:	fuente "backup" de ED	$$$:	fichero temporal del ED
BAS:	fuente BASIC	PAS:	fuente PASCAL
COB:	fuente COBOL	FOR:	fuente FORTRAN
REL:	módulo objeto reubicable	INT:	módulo objeto intermedio

1.3.1.8. Los caracteres de control de edición

El módulo CCP permite además gestionar la "aparición de datos" en el terminal tratando ciertos caracteres de control del teclado de forma que se correspondan con determinadas funciones de captura de datos o de edición.

<rubout> :	borrado del carácter precedente;
<ctrl-H> :	borrado del carácter precedente (en CP/M 2.2);
<ctrl-U> :	borrado de una línea completa;
<ctrl-X> :	igual que <ctrl-U>
<ctrl-R> :	reimprime la línea actual en forma correcta;
<crtl-M> :	fin de línea (retorno de carro: <CR>;
<ctrl-J> :	alimentación de línea (line-feed: <LF>);
<ctrl-E> :	salto de la siguiente línea sin transmisión de la línea al CCP;
<ctrl-I> :	activación de la tabulación, por bloques de 8 columnas;
<ctrl-Z> :	fin de fichero (utilizado por ED y PIP);
<ctrl-C> :	rearranque del sistema por una llamada al "warm start";
<crtl-P> :	eco de las entradas-salidas sobre la impresora; funciona de forma alternativa (validación e invalidación);
<ctrl-S> :	elimina la visulización sobre la consola.

1.3.1.9. Los mandatos integrados del CCP

Además de las funciones de diálogo, el CCP contiene cinco o seis mandatos residentes de utilización muy corriente, cuyo funcionamiento explicaremos más adelante:

DIR	:	impresión de los nombres de los ficheros del directorio
ERA	:	destrucción de un fichero
REN	:	cambio del nombre de un fichero
SAVE	:	salvaguarda de la memoria en un fichero
TYPE	:	listado de un fichero (ASCII) sobre la consola
USER	:	cambio del número de usuario (versión CP/M 2.2)

1.3.2. El BDOS

La misión del BDOS consiste en ofrecer al usuario una interfase flexible y poderosa para la gestión de las informaciones existentes en las memorias masivas. Proporciona al usuario todas las funciones preprogramadas necesarias para la manipulación de ficheros y de registros. Desde el punto de vista interno, y por lo tanto transparente para el usuario, el BDOS gestiona el espacio de almacenamiento de los ficheros en disco.

1.3.2.1. Estructura física de los ficheros

Un fichero está constituido por una serie secuencial de registros de 128 octetos cada uno (tamaño del sector físico). Los ficheros en CP/M pueden alcanzar un tamaño de 65536 registros de 128 octetos, es decir, una capacidad máxima de 8 Mega-octetos. Estos registros se consideran, desde el punto de vista "lógico", como si estuviesen consecutivos aunque no lo estén necesariamente en el soporte "físico".

El CP/M ofrece dos métodos de acceso a los registros de los ficheros: el secuencial y el acceso directo.

Desde el punto de vista interno, y por lo tanto invisible para el usuario, los ficheros se dividen en segmentos de 16 K-octetos, llamados "extensiones lógicas".

Por razones de comodidad para el acceso a los registros, éstos se reagrupan en una estructura lógica llamada "bloque". El bloque representa la unidad de almacenamiento (operaciones READ y WRITE).

El tamaño de un bloque es fijo para un sistema dado, pero puede redefinirse durante la generación de otro sistema, modificando la tabla de parámetros del disco. Este tamaño puede tomar los valores de 1, 2, 4, 8 ó 16 Kilo-octetos. Por ejemplo, en el caso de un disquete en el que el tamaño del bloque es 2K, un bloque podrá contener de 1 a 16 registros de 128 octetos.

La asignación de los bloques se hace dinámicamente, conforme a la demanda. Los bloques que constituyen un fichero no tienen que estar forzosamente contiguos en el disco: pueden existir "huecos" debidos a liberaciones anteriores de bloques (destrucción de un fichero, por ejemplo).

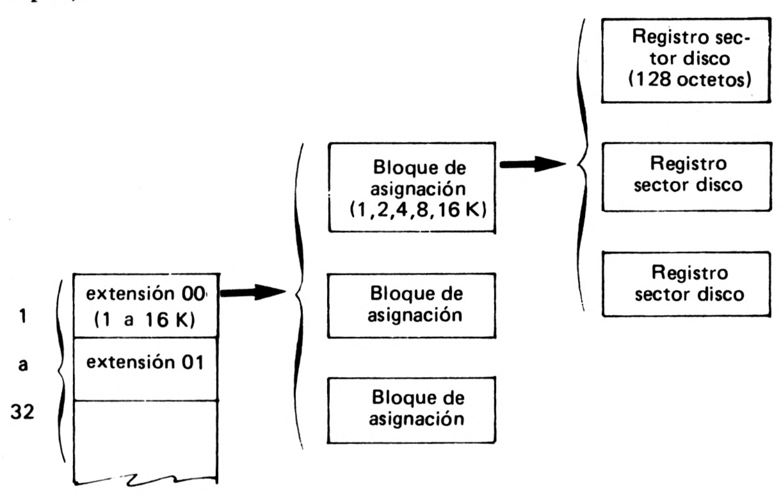


Fig. 3.– Estructura de un fichero en disco.

1.3.2.2. Organización lógica de los ficheros

No existe, hablando con propiedad, una estructura de los ficheros como la que encontramos habitualmente en los grandes sistemas. Sin embargo, gracias a los dos sistemas de acceso, algunos usuarios pueden organizar sus ficheros para tener un acceso secuencial indexado (ISAM), por ejemplo COBOL. Así pues, al usuario se le deja una total libertad de organización a nivel de fichero.

Asimismo, el CP/M no impone de forma rigurosa una estructura de registro, excepto para los ficheros tipo "texto" en ASCII en los que el final del registro viene señalado por los dos caracteres de función <CR> (retorno de carro) y <LF> (line-feed).

El final de un fichero tipo "texto" viene indicado por el carácter <ctrl-Z> o 1AH (en hexadecimal), o por un final real de fichero, es decir, ya no hay más sectores después de una operación de lectura de registros.

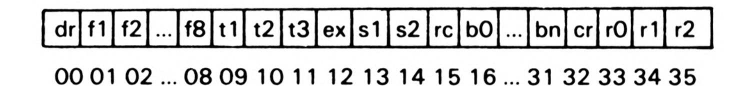
1.3.2.3. El bloque de control del fichero FCB

Desde el punto de vista externo, es decir, el del usuario, un fichero está identificado por su nombre, su tipo y el nombre del disco en el que se encuentra.

Desde el punto de vista interno, es decir, el del sistema, un fichero está asociado a un cierto número de parámetros: dirección física de los bloques de asignación, contadores de registros...

Todas estas informaciones, externas e internas, que caracterizan totalmente al fichero, están agrupadas en un área de memoria llamada FCB (File Control Block). El tamaño de un FCB es de 32 octetos para los ficheros de acceso secuencial y de 36 octetos para los ficheros de acceso directo. No existe ninguna limitación sobre el número de FCB's utilizables en un programa. Existe un FCB por defecto, gestionada normalmente por el CCP, en la dirección 005CH.

La estructura del FCB es la siguiente:



dr	número del disco (0-16) 0 → disco actual tomado por defecto 1 → disco A 2 → disco B ... 16 → disco P
f1...f8	nombre del fichero en ASCII mayúsculas
t1, t2, t3	tipo del fichero, opcionalmente completado a blancos
ex	número actual de la extensión del fichero, inicialmente 0
s1, s2	reservados para el BDOS
rc	contador de registros por extensión (0-128)
b0...bn	relación de la asignación en disco de los bloques del fichero
cr	número del registro actual en modo secuencial
r0, r1, r2	número de registro, opcional en modo directo (0-65535)

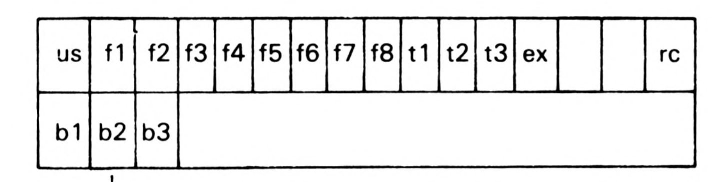
1.3.2.4. Concepto de directorio de ficheros

Para cada fichero en uso, el BDOS gestiona un directorio que contiene el nombre del fichero y las informaciones necesarias para su acceso (tamaño en bloques, dirección de cada bloque, etc...).

Cada entrada de este directorio (128 entradas como máximo) ocupa 32 octetos. Los 16 primeros octetos son una copia exacta de los 16 primeros octetos del FCB en el momento del cierre del fichero, nombre del fichero, tipo del fichero, y contador de registros con la excepción de que el primer octeto ya no expresa el número de disco asociado al fichero, ya que el directorio es único para ese disco, sino el número del usuario (ver mandato USER bajo CP/M 2.2).

Los 16 octetos siguientes representan el "MAPA" (MAP) de ocupación del disco por el fichero. A cada octeto del "MAPA" le corresponde un número de bloque de nK-octetos, dependiendo de la generación del sistema. El propio directorio ocupa uno o dos bloques según las configuraciones de discos. El primer fichero en disco quedará contiguo al directorio y la numeración empezará a partir de 1 ó 2 según la configuración.

Estructura de una entrada del directorio:



Números relativos en relación al final de directorio de los bloques de ubicación en disco del fichero afectado.

Las búsquedas en el directorio se realizan mediante comparaciones sucesivas de cada una de las entradas del directorio con el FCB del usuario sobre el campo simbólico: nombre del fichero.

1.3.2.5. Funciones del BDOS

El BDOS proporciona al usuario, y a las otras partes integrantes del sistema, numerosas rutinas de servicio como:

- la re-inicialización del CP/M
- la selección de un disco
- la creación, apertura y cierre de un fichero
- la lectura o escritura de un registro
- la búsqueda de una entrada en el directorio de un disco
- la creación y la destrucción de entradas en el directorio

– el cambio de nombre de un fichero
– la inicialización de la dirección DMA (acceso directo memoria)
– la lectura de vectores (discos activos, disco actual)
– la protección temporal contra escritura de un disco
– el posicionamiento de un atributo de fichero (protección)
– lectura, escritura, posicionamiento de registros en modo directo.

1.3.3. El BIOS

El BIOS contiene el conjunto de los programas de entradas-salidas (drivers) que son específicos de la configuración física adoptada.

1.3.3.1. Adaptabilidad

El conjunto de estos programas ligados al ambiente externo está generalmente concebido y escrito por el constructor del micro-ordenador. En principio, al comprador se le proporciona el "fuente" del BIOS escrito en lenguaje máquina (ensamblador o macro-ensamblador) ya que es posible que éste quiera modificar, borrar o añadir determinados programas en función de sus propias necesidades (por ejemplo, entrada-salidas sobre periféricos no estándares).

Es posible que sea ésta una de las razones por la que el sistema CP/M es tan fácilmente adaptable a una configuración dada, aunque para ello se necesite el conocimiento en profundidad del funcionamiento de los elementos de entrada-salida, así como cierta experiencia en la programación en ensamblador.

Aunque el BIOS es un módulo independiente de los otros módulos (CCP y BDOS) les proporciona las primitivas de entradas-salidas indispensables para su funcionamiento.

1.3.3.2. Interfase

Una interfase, definida por unas especificaciones muy concretas, entre el BIOS y el resto del sistema, permite establecer el enlace entre los otros módulos del sistema: el CCP y el BDOS.

Esta interfase normalizada está constituida por una secuencia de 17 instrucciones de bifurcación hacia las funciones específicas. La secuencia de "JMP" llamada "jump vector" (vector de bifurcación) está situada en la base del BIOS. Observemos que los parámetros de las se-

cuencias de llamada al BIOS, así como el retorno de los resultados, deben respetar estrictamente las especificaciones impuestas por Digital Research.

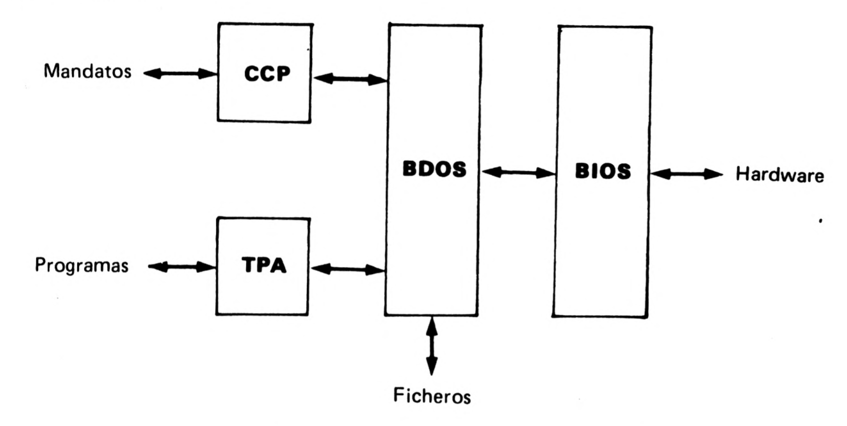


Fig. 4.— Arquitectura de los módulos del CP/M.

1.4. ESTRUCTURA DEL SISTEMA CP/M

1.4.1. Estructura de la memoria

La memoria bajo CP/M está dividida en cinco áreas, como se muestra en la figura que aparece a continuación. El cargador específico del sistema coloca los tres módulos del CP/M en las posiciones altas de la memoria. El CCP está referenciado por la dirección CBASE, el BDOS por la dirección BIAS y el BIOS que ocupa hasta el final de la memo-

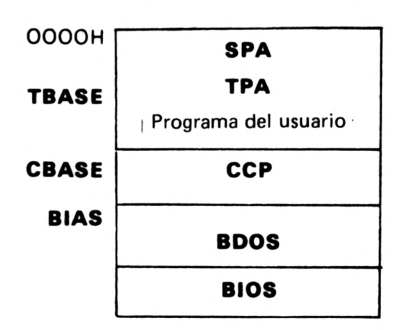


Fig. 5.— Implantación de memoria de los módulos de CP/M.

ria. Las posiciones bajas de la memoria, desde la 0000H hasta la 00FFH (es decir, 256 octetos) están reservadas para el sistema. Por último, el área llamada "TPA", que comienza a partir de la dirección TBASE, es el área que queda disponible para los programas del usuario.

1.4.1.1. La TPA

La TPA (Transient Program Area) es la parte más grande de memoria disponible para la ejecución de los programas. Estos programas pueden ser programas del usuario o incluso programas del sistema no residentes de tipo "COM".

Este área está situada entre las direcciones 0100H (hexadecimal) y CBASE (base del CCP). En el caso de algunos micro-ordenadores que poseen determinados programas pre-grabados en PROM, la dirección de comienzo de la TPA se traslada a la posición 4300H en lugar de la 0100H. El tamaño de la TPA es ampliable según la configuración de memoria adoptada: 16, 32, 48 ó 64K. La dirección de comienzo (TBASE = 0100H ó 4300H) se mantiene fija; lo que varía, en múltiplos de 16K (4000H), es la dirección de CBASE.

Además el CP/M ofrece la posibilidad de cargar grandes programas que ocupen más allá de la dirección CBASE, solapándose con el CCP. Mediante el "warm start", desencadenado cuando finaliza la ejecución del programa, o mediante la intervención del usuario (tecleando el carácter especial <ctrl-C>) se volverá a cargar automáticamente el CCP en memoria.

1.4.1.2. La SPA

La SPA (System Parameter Area) es el área reservada del sistema CP/M para albergar una serie de parámetros y vectores de bifurcación.

A continuación se detalla la ubicación del código y los datos contenidos en este área:

Direcciones	**Descripción de los contenidos**
0000H–0002H	Contiene una instrucción JMP hacia el punto de entrada del "warm-start" (re-inicialización en caliente) permitiendo al usuario la programación de un retorno monitorizado (JMP 000H) cuando finalice la ejecución de su programa.

0003H–0003H	Contiene el octeto IOBYTE que permite la reasignación de los periféricos. El octeto IOBYTE está formado por cuatro campos diferentes, de dos bits cada uno, llamados CONSOLE, READER, PUNCH, LIST: 0003H [LIST | PUNCH | READER | CONSOLE] bits 7,6 bits 5,4 bits 3,2 bits 1,0 El contenido de cada campo puede tomar cuatro valores para definir la asignación origen y destino de cada periférico lógico: TTY:, CRT:, RDR:, PUN:, LPT:, BAT:, UC1:, UL1:, UP1:, UR1:, etc... Observemos que la implantación de la función IOBYTE es opcional y afecta solamente a la organización del módulo BIOS.
0004H–0004H	Contiene el número del disco actual (0 = A, ... 15 = P)
0005H–0007H	Contiene una instrucción JMP hacia el punto de entrada del BDOS que permite llamar a las funciones monitor o conocer cuál es la última dirección disponible para los programas de usuarios (LHLD 0006H).
0008H–003FH	Destinado a las interrupciones. La interrupción 7 (JMP en 0038H) es utilizada por los programas "debug" DDT y SID (paradas en dirección, trace).
0040H–004FH	Area de 16 octetos reservada para el BIOS. No utilizada bajo CP/M estándar.
0050H–005BH	Area reservada de poca utilización.
005CH–007CH	Bloque de control del fichero (FCB) por defecto, suministrado al programa por el CCP.
0080H–00FFH	Zona tampón implícita de 128 octetos reservada para las entradas-salidas de disco y consola.

1.4.2. Estructura del disco

Para cargar un sistema CP/M en memoria ("cold start" o arranque en frío), el disco del sistema debe encontrarse en la "unidad" A. Este disco contiene el sistema CP/M propiamente dicho, el directorio asociado al disco y los ficheros en el directorio.

El mandato DIR permite listar los nombres de los ficheros presentes en el disco.

```
A>DIR
A: MOVCPM  COM : SUBMIT   COM : ED       COM  : ASM      COM
A: PIP     COM : LOAD     COM : STAT     COM  : SYSGEN   COM
A: DDT     COM : MODE     COM : DUMP     ASM  : CBIOS    ASM
A: DUMP    COM : CBIOS64  COM : CBIOS48  COM  : CBIOS32  COM
A>
```

1.4.2.1. Ubicación del CP/M: las pistas reservadas

El sistema CP/M residente en disco no está representado con el formato de un fichero estándar, ni siquiera como un fichero especial. Se comprueba de hecho que aunque se destruyan todos los ficheros existentes en el disco, la operación de "bootstrap" del sistema (arranque en frío o "cold start") funciona con toda normalidad. En ese momento, el mandato DIR no visualizará ningún nombre de fichero.

```
A>DIR
NO FILE
A>
```

Como medida de protección, el sistema CP/M no aparece a nivel de usuario, aunque se encuentre presente en el disco.

Las dos primeras pistas del disco están reservadas para los módulos del sistema operativo propiamente dicho. Cada pista (numerada a partir de 0), contiene 26 sectores (numerados a partir de 1), para un disquete de simple densidad.

El sector 1 de la pista 0 contiene el "bootstrap" o "cold start loader" cuya misión es la de cargar el resto del sistema (CCP, BDOS, BIOS) en memoria a partir de la dirección CBASE.

La dirección CBASE se calcula, en función del espacio disponible, de la siguiente manera:

Capacidad	CBASE
16 K	2400H
32 K	6400H
48 K	A400H
64 K	E400H

Las asignaciones en disco y en memoria de los diferentes módulos del sistema CP/M, así como su correspondencia, aparecen en el esquema siguiente:

Pista	Sector	Dirección de memoria	Módulo CP/M
00	01	Dirección del boot	BOOT
00 .. 00	02 .. 17	CBASE .... CBASE+0780H	CCP
00 .. 01	18 .. 19	CBASE+0800H .... CBASE+1580H	BDOS
01 .. 01	20 .. 26	CBASE+1600H .... CBASE+1900H	BIOS
02 .. 76	01 .. 26		Directorio y fichero

Fig. 6.– Relación disco-memoria de los módulos CP/M.

1.4.2.2. Ubicación del directorio y de los ficheros

El directorio del disco (o directory) empieza en la pista 2, sector 1. Ocupa 32 sectores como máximo, divididos en zonas de 32 octetos llamadas "entradas", es decir, cuatro entradas por sector o también 128 entradas por disco. Una entrada es la copia de los 32 primeros octetos del FCB después del cierre del fichero (close).

La búsqueda de un fichero se realizará mediante un barrido del directorio, para comparar cada entrada con el FCB del usuario.

La zona de almacenamiento de los ficheros de usuario empieza a partir del final del directorio.

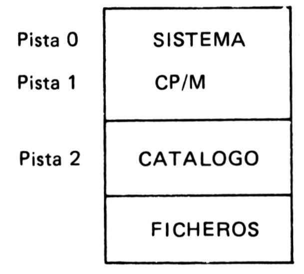


Fig. 7.– Estructura de un disco del sistema.

1.5. LAS FACILIDADES BAJO CP/M: LOS MANDATOS

En esta sección del libro vamos a dejar las generalidades para situarnos a nivel del usuario delante de su terminal. Ya hemos visto que sólo se pueden teclear los mandatos cuando previamente ha aparecido en pantalla el carácter de invitación llamado "prompt". Un mandato también puede presentar varios argumentos en la misma línea como nombres de ficheros con algún tipo de similitud, o valores diversos.

Existen dos tipos de mandatos bajo CP/M: los mandatos integrados al sistema en el CCP, también llamados residentes, y los mandatos estándares (suministrados con el CP/M) asociados a unos ficheros ejecutables del tipo "COM" Normalmente estos ficheros se encuentran en el mismo disco que contiene al sistema, por razones de homogeneidad.

1.5.1. Los mandatos residentes

Se trata de mandatos de interés general y de utilización muy frecuente. Además, debido a su reducido tamaño, se ha podido integrarlos con el CCP, en la memoria central.

1.5.1.1. DIR

El mandato DIR permite visualizar el nombre de los ficheros presentes en el disco seleccionado y que satisfacen la referencia ambigua dada como argumento.

Ejemplos de mandatos DIR correctos:

A > DIR	lista el nombre de todos los ficheros presentes en el disco actual; es equivalente al mandato: "DIR *.*".
A > DIR B:	lista el nombre de todos los ficheros presentes en el disco B.
A > DIR *.COM	lista el nombre de todos los ficheros de tipo COM.
A > DIR PRU???.B?S	lista el nombre de los ficheros que satisfagan esta referencia ambigua. Ejemplo: "PRUEBA.BAS".
A > DIR B:PRUEBA.*	lista el nombre de los ficheros genéricos del fichero "PRUEBA" presentes en el disco B.

Si no se encuentra ningún fichero en el disco seleccionado se visualiza el mensaje "NO FILE" o "NOT FOUND" en la consola.

1.5.1.2. ERA

El mandato ERA (Erase) permite destruir uno o varios ficheros del disco seleccionado. Los ficheros que se van a suprimir deben satisfacer la referencia de fichero dada como argumento en la misma línea del mandato.

De hecho, los ficheros afectados no se borran físicamente del soporte, simplemente se les deja inaccesibles. En el primer octeto de la entrada, en el directorio, del fichero que se va a suprimir se escribe el valor E5H (en hexadecimal). El hecho de transformar esta entrada permite ignorar el nombre del fichero y su "MAP" de ocupación de disco; el espacio ocupado se volverá a utilizar cuando se haga uso de esta entrada para la creación de otro fichero.

Los ejemplos que damos a continuación aclaran la utilización del mandato ERA:

A>ERA ABC.XYZ	suprime el fichero "ABC.XYZ"
A>ERA *.BAK	suprime todos los ficheros de tipo BAK.
A>ERA TEST.*	suprime todos los ficheros cuyo nombre principal es "TEST".
A>ERA B:EX.ASM	suprime el fichero "EX.ASM" del disco B.

A>ERA *.* suprime todos los ficheros del disco actual. Esta operación puede ser fatal si el usuario ha tecleado este mandato por error. Por este motivo se visualiza el mensaje de advertencia:

"ALL FILES (Y/N)?"

en la consola para solicitar la confirmación por parte del usuario.

1.5.1.3. REN

El mandato REN (Rename) permite cambiar el nombre de un fichero ya existente. Las referencias ambiguas no tienen sentido en este tipo de mandato y no están autorizadas. La sintaxis es la siguiente:

REN fichero1 = fichero2

"fichero1" representa el nuevo nombre del fichero y "fichero2" el antiguo. Estos nombres explícitos son del tipo: "D:FFFFFFFF.TTT" donde "D:" representa el nombre del disco (opcional), "FFFFFFFF" el nombre principal del fichero (obligatorio) y ".TTT" el tipo (opcional).

Ejemplos de mandatos REN:

```
A>REN A:NUEVO.MAC = A:VIEJO.ASM
A>REN A:DESPUES.TXT=ANTES
```

Si el fichero que se va a renombrar no existe, el sistema devuelve el mensaje "NOT FOUND". Asímismo, si el nuevo nombre corresponde a un fichero ya existente, se visualiza en consola el mensaje "FILE EXISTS".

1.5.1.4. SAVE

El mandato SAVE permite salvar en disco los programas cargados en memoria, en el área TPA, con el formato de un fichero. La sintaxis del mandato SAVE es la siguiente:

SAVE n fichero

El primer argumento "n" representa el número decimal de páginas de memoria a salvar a partir del comienzo del área TPA. Una página de

memoria es equivalente a un bloque contiguo de 256 octetos (100H en hexadecimal). El cálculo de "n" se hace de la forma siguiente: conociendo la dirección de la última posición del programa cargado en memoria (ver LOAD, DDT) se obtiene el tamaño en octetos restando 100H de la dirección final y si el resultado no es múltiplo de 100H se le añade 1 a la cifra de las centenas y se convierte este número en decimal.

Ejemplo: supongamos que un programa termina en la dirección 12C8H; su tamaño es 12C8H–100H=11C8H y el número de páginas es entonces 11+1=12H, es decir, n = 18 en decimal.

El segundo argumento del mandato representa el nombre del fichero en el que se guardará la imagen en memoria del programa. En general, este fichero es ejecutable y su tipo debería ser "COM".

Ejemplos:

A>SAVE 5 PROG.COM	salva el área TPA, desde 100H hasta 5FFH, en el fichero "PROG.COM" sobre el disco actual.
A>SAVE 12 B:SALVAR	salva el área TPA, desde 100H hasta 0CFFH, en el fichero "SALVAR" en el disco B.

1.5.1.5. TYPE

El mandato TYPE permite visualizar el contenido de un fichero "fuente" (source) codificado en ASCII. El nombre dado como argumento debe ser una referencia explícita. Los ficheros de tipo ASCII están organizados generalmente como una serie de elementos separados por 0DH, 0AH (retorno de carro, line-feed). TYPE también extiende los tabuladores horizontales, especificados en el fichero con el octeto 09H o <ctr l-I>, a ocho espacios separadores. Los ficheros que no son de tipo ASCII puedan dar resultados imprevisibles en la visualización de su contenido, dependiendo del tipo de pantalla conectada al micro-ordenador.

Ejemplos de utilización de TYPE:

A>TYPE SOURCE.COB	lista un fichero fuente escrito en COBOL
A>TYPE LISTING.PRN	lista el fichero imagen de la impresora.
A>TYPE B:CALCULO.FOR	lista el fichero fuente "CALCULO" escrito en FORTRAN sobre el disco B.

1.5.2. Mandatos estándares no residentes

El conjunto de estos mandatos viene incluido con el producto CP/M en un disquete de sistema "bootable". Estos mandatos, como ya hemos visto, son unos ficheros tipo "COM". Por supuesto, estos mandatos no están ligados al disquete de sistema y se les puede ejecutar desde otros disquetes o desde discos duros.

Algunos de estos mandatos poseen, a su vez, unos sub-mandatos. Nos encontramos entonces con unas "utilidades" que entablan su propio diálogo con el usuario, éste es el caso de ED y DDT.

1.5.2.1. STAT

El mandato STAT proporciona al usuario ciertas informaciones acerca de los ficheros como el espacio de almacenamiento o las asignaciones de los periféricos. Ofrece, además, la posibilidad de cambiar las asignaciones de los periféricos y los atributos de los ficheros. Este mandato admite unos argumentos, en la misma línea, que son funciones de las informaciones que desea obtener el usuario.

Informaciones sobre los soportes en disco

En este caso el argumento es el nombre del disco (A:,...P:); si se omite, se toma por defecto el conjunto de discos presentes en la configuración.

Sintaxis posibles:

STAT Calcula el espacio disponible en los discos presentes actualmente en el sistema e imprime el mensaje:

```
d:R/W,SPACE:xxxK
```

o

```
d:R/O,SPACE:xxxK
```

R/W indica que se puede leer (Read) y escribir (Write) en el disco "d:", mientras que R/O indica que el disco "d:" está protegido contra escritura (Read Only), ya sea después de una demanda explícita, ya sea después de una carga del disquete en caliente. El valor "xxxK" da el tamaño del espacio que todavía queda disponible en el disquete, en Kilo-octetos.

STAT d: Calcula únicamente el espacio disponible en el disco "d:" seleccionado y visualiza el mensaje:

BYTES REMAINING ON d:xxxK

Informaciones sobre los ficheros

El argumento que va a continuación de STAT puede ser o una referencia explícita o una referencia ambigua de nombre de fichero.

STAT "nombre de fichero

Los nombres de los ficheros que satisfagan la referencia aparecen listados por orden alfabético, con sus características de almacenamiento. Una línea de cabecera explicita los campos de información visualizados.

Bajo CP/M 1.4:

```
RECS   BYTS   EX   D:FILENAME.TYP
rrrr   bbbK   ee   d:ffffffff.ttt
```

El campo "RECS" da el número de registros "rrrr" de 128 octetos de que consta el fichero. El campo "BYTS" da el número de bloques asignados al fichero en kilo-octetos: (bbb = rrr * 128/1024). El campo "EX" da el número de extensiones de 16K del fichero (ee=bbb/16). A continuación viene el nombre del fichero perteneciente al disco "d".

Al final del listado aparecen las características individuales de los ficheros que responden al criterio de selección así como el espacio que todavía queda disponible.

Bajo CP/M 2.2:

```
Recs   Bytes   Ext   Acc
rrrr   bbbK    ee    R/W d:ffffffff.ttt
```

Los campos "Recs", "Bytes" y "Ext" son similares a los campos "RECS", "BYTS" y "EX" de la versión 1.4. El campo "Acc" da, además, el modo de acceso a los ficheros, el cual puede ser modificado mediante los mandatos de cambio de los atributos de fichero, especificados para la versión 2.2.

STAT d:fichero.tip $R/W	se permite el acceso para lectura/escritura al fichero especificado.
STAT d:fichero.tip $R/O	protección del fichero especificado contra las operaciones de escritura y borrado.
STAT d:fichero.tip $SYS	indicador "sistema" para hacer que no figure el nombre del fichero después del mandato DIR.
STAT d:fichero.tip $DIR	anula el efecto del indicador sistema "SYS" para volver a dejar el fichero en su situación normal.

Una vez que un fichero se ha marcado con el modo R/O, se rechazará cualquier intento de borrado del fichero o de escritura en él. En la pantalla se visualiza el mensaje:

Bdos Err on d:fichero R/O

y el sistema se bloquea (se queda a la espera de un carácter). Al pulsar cualquier tecla se provoca un "warm start" así como el desbloqueo del sistema.

Asignación de periféricos

Como en la mayoría de los sistemas operativos modernos, el acceso a un elemento periférico no se realiza directamente. En particular, los compiladores u otros programas evolucionados no reconocen a los periféricos físicamente presentes en el sistema, sino a unas unidades llamadas lógicas que pueden quedar afectadas por ellos.

Por ejemplo, si la impresora está averiada, o no está conectada, el usuario puede reasignar esta impresora a la consola principal para visualizar sus resultados o incluso direccionar la salida a impresión hacia otro periférico según su elección.

El CP/M sabe cómo gestionar estas unidades lógicas permitiendo así una gran flexibilidad de utilización a nivel de dispositivos de entradas-salidas.

El octeto IOBYTE que se encuentra en la posición 0003H de la memoria, contiene las correspondencias entre las unidades lógicas y las unidades físicas (ver capítulo sobre la SPA).

Con la excepción de los discos, existen cuatro periféricos asignables bajo CP/M. Estos periféricos tienen nombres físicos y lógicos predefinidos.

Con el mandato: "STAT. VAL:" (versión CP/M 1.4) se obtiene la lista de las asignaciones posibles:

Ejemplo:

```
A>STAT VAL:
CON : = TTY:  CRT :  BAT :  UC1:
RDR : = TTY:  PTR :  UR1 :  UR2:
PUN : = TTY:  PTP :  UP1 :  UP2:
LST  : = TTY:  CRT:  LPT :  UL1:
A>
```

donde los nombres de los periféricos lógicos son:

CON:	la consola de diálogo con el usuario.
RDR:	una lectura magnética (casete) o de papel
PUN:	una perforadora de cinta (papel) o una grabadora (casete).
LST:	una impresora para salida de listados.

y los nombres de los periféricos físicos:

TTY:	teletipo (baja velocidad)
CRT:	consola, pantalla (Cathode Ray Tube).
BAT:	dispositivo para tratamiento por lotes (batch).
PTR:	lectora de cinta de papel o de casete magnética.
PTP:	perforadora de cinta de papel o grabadora de casete magnética.
LPT:	impresora.
UC1:	otro dispositivo de tipo consola.
UR1:	otro dispositivo de tipo lectora.
UR2:	otro dispositivo de tipo lectora.
UP1:	otro dispositivo de tipo grabadora/perforadora.
UP2:	otro dispositivo de tipo grabadora/perforadora.
UL1:	otro dispositivo de tipo impresora.

El mandato "STAT VAL:" (versión CP/M 2.2) imprime el resumen de los mandatos disponibles.

Ejemplo:

```
A>STAT VAL:
Temps R/O Disk: d:=R/O
Set Indicator    : d:filemane. typ $R/O $R/W $SYS $DIR
```

```
Disk Status    : DSK: d:DSK:
User Status    : USR:
Iobyte Assign:
CON  := TTY:  CRT :  BAT :  UC1:
RDR  := TTY:  PTR :  UR1 :  UR2:
PUN  := TTY:  PTP :  UP1 :  UP2:
LST  := TTY:  CRT :  LPT :  UL1:
A>
```

La sintaxis de la asignación de un periférico físico a un periférico lógico es de la forma:

STAT lógico = físico

Ejemplo:

A>STAT CON:=CRT:, LST:=LPT:

En cualquier momento se pueden visualizar las asignaciones que están activas. En particular si se desea conocer cuáles son las asignaciones por defecto, el usuario deberá teclear el mandato "STAT DEV:".

Ejemplo:

```
A>STAT DEV:
CON: is CRT:
RDR: is PTR:
PUN: is PTP:
LST : is LPT:
```

Protección de un disco

Siempre que lo desee el usuario puede proteger un disco para evitar la destrucción de la información que contiene, en caso de un posible accidente. La sintaxis es la siguiente:

STAT d:=R/O

El disco queda entonces protegido contra escritura y cualquier intento de acceso, hasta la siguiente desprotección o hasta el siguiente "warm start", se resuelve con la presentación del mensaje de error:

```
BDOS ERR ON d: READ ONLY
```

Mandatos anexos bajo CP/M 2.2

```
STAT USR:
```

da la lista de los números de los usuarios (ver mandato USER) que poseen ficheros en el disco actual:

Active User	: u	número del usuario activo.
Active file	: u1 u2 ... un	números de los usuarios que poseen ficheros sobre el disco actual.

El mandato:

```
STAT DSK: o STAT d:DSK:
```

da las características físicas y lógicas del disco seleccionado.

Ejemplo:

A>STAT DSK:

```
   A : Drive Characteristics
1944 : 128 Byte Record Capacity
 243 : Kilobyte Drive Capacity
  64 : 32 Byte Directory Entries
  64 : Checked Directory Entries
 128 : Records/Extent
   8 : Records/Block
  26 : Sectors/Track
   2 : Reserved Tracks
```

Así pues este disco tiene una capacidad de 1.944 registros de 128 octetos, es decir, 243 Kilo-octetos. El directorio del fichero compren-

de un máximo de 64 posibles entradas que han sido verificadas. Una extensión del fichero es equivalente a 128 registros y un bloque corresponde a ocho registros. La sectorización del disco es de 26 sectores por pista, lo cual corresponde generalmente a un disco formateado en simple densidad. Para el sistema CP/M propiamente dicho se han reservado dos pistas.

1.5.2.2. PIP

PIP (Peripheral Interchange Program, programa de intercambio entre periféricos), es un potente programa de utilidad para la manipulación de ficheros o la conversión de soporte en las unidades periféricas. PIP acepta dos tipos de sintaxis:

```
PIP
```

o

```
PIP <mandato interno del PIP>
```

En el primer caso, PIP invita al usuario a que teclee sus mandatos línea a línea, imprimiendo el carácter de diálogo "*". Se sale del mandato PIP introduciendo una línea en blanco, es decir, un retorno de carro <cr>.

Ejemplo:

```
A>PIP
* mandato interno del PIP
* mandato interno del PIP
* <cr>
A>
```

En el segundo caso, PIP no acepta más que un solo mandato en la misma línea. El mandato se ejecuta inmediatamente y se devuelve el control al sistema al final de su ejecución.

En los dos casos la sintaxis del mandato interno del PIP es idéntica; su forma general es la siguiente:

```
destino = fuente1, fuente2, ..., fuenten
```

El primer campo "destino" representa el fichero o el periférico que va a recibir los datos. Los campos "fuente" representan uno o varios ficheros o periféricos desde los que se copiarán los datos.

Si se han indicado varios ficheros fuente, éstos deben ser de tipo ASCII y terminar con el carácter de fin de fichero<crt l+Z>.Una línea de mandato no debe contener más de 255 caracteres.

Los argumentos "destino" y "fuente" pueden ser referencias ambiguas precedidas del nombre del disco, como ya hemos dicho anteriormente.

Copia de uno o varios ficheros

La sintaxis de copia de fichero es un caso particular de la sintaxis global que acabamos de ver. Este mandato no necesita más que un solo campo "fuente":

destino = fuente

Solamente puede ser ambiguo el campo "fuente". Además PIP permite más mandatos abreviados para la transferencia de ficheros entre discos. En este caso aparece el nombre del disco en uno de los dos campos, el fichero "destino" toma entonces el mismo nombre que el del fichero "fuente";

PIP d: = fuente	copia en el disco "d:" el/los ficheros del disco actual que satisfacen la referencia "fuente".
PIP d:=s:fuente	copia en el disco "d:" el/los ficheros del disco "s:" que satisfacen la referencia "fuente".
PIP destino=s:	copia en el disco actual el fichero "destino" del disco "s:".
PIP d:destino=s:	copia el fichero "destino" del disco "s:" en el disco "d:".

Ejemplos de mandatos PIP correctos:

A>PIP COPIE=ORIGINAL

A>PIP B:NEW.DOC=A:OLD.TEX

B>A:PIP B:=A:MBASIC.COM

A>PIP B:=*.BAS

Copia de disco a disco

El mandato PIP permite la copia de todos los ficheros de un disco sobre otro disco; no permite la copia del propio sistema CP/M (ver el mandato de generación del sistema SYSGEN). La copia de todo un disco resulta muy útil a menudo para archivar o salvar los programas o los datos ya que los soportes de disco son relativamente frágiles. En particular los disquetes tienen una duración de vida limitada, ya que las cabezas están en contacto físico con el soporte.

A>B:=A:*.*   copia todos los ficheros de A a B

También es frecuente encontrar, junto con los productos estándares del CP/M, unos programas de utilidad para la copia integral de disquetes en simple o en doble densidad:

COPY	copia de disquetes en simple densidad
DCOPY	copia de disquetes en doble densidad

Concatenación de ficheros

La sintaxis de la concatenación responde a la sintaxis general del mandato PIP. Los ficheros "fuente" de tipo ASCII se leen en el orden dado por su definición y se reagrupan en un único fichero destino.

Ejemplo:

A>PIP TOTAL.ASM=PART1.ASM, PART2.ASM, B:PART3.ASM

Conversión de soportes

PIP permite la transferencia de datos de un fichero a un periférico y viceversa. Los nombres de los periféricos lógicos y físicos son los ya definidos en el mandato STAT. PIP reconoce también un cierto número de periféricos adicionales como:

NUL: emite 40 caracteres nulos (00H) hacia el dispositivo de salida (trozo de cinta perforada, por ejemplo).

EOF: marca de fin de fichero (1AH o <ctrl-Z> en el dispositivo de destino.

INP: entrada especial para un periférico no estándar. PIP hace un "CALL" a la dirección 103H que es donde el usuario ha introducido su rutina de lectura modificando PIP mediante DDT. El carácter leído se devuelve en la posición 109H.

OUT: salida especial para un periférico no estándar. PIP hace un "CALL" a la dirección 106H que es donde el usuario ha introducido su rutina de escritura. El carácter que se va a emitir debe encontrarse en el registro C.

PRN: nombre lógico para una salida de tipo impresora teniendo en cuenta los tabuladores, saltos de página y la numeración de las líneas.

Los mandatos PIP aceptan tanto nombres de ficheros como unidades lógicas de periféricos. En ambos casos los datos "fuentes" se leen hasta que se detecta el final de fichero, <ctrl-Z> para los ficheros ASCII y un fin real de fichero para los ficheros no ASCII en disco (no quedan más sectores). El final de fichero <ctrl-Z> se añade al fichero destino una vez que se ha completado la transferencia. Si el fichero destino ya existe, se destruye, y se crea de nuevo a partir del fichero fuente especificado.

Pulsando una tecla cualquiera del teclado, se puede abandonar la fase de copia, en cualquier momento. PIP visualiza el mensaje "ABORTED" para indicar que la operación de transferencia ha terminado. Del mismo modo, si se produce un error durante el proceso, PIP abandona todos los mandatos que queden en cola (si los hay) y devuelve el control al sistema.

Ejemplos de mandatos de conversión entre soportes:

A>PIP LST: = LISTING.PRN	copia el fichero LISTING.PRN sobre el periférico asignado a LST:.
A>PIP B:LEER.CIN = PTR	copia en el fichero LEER.CIN los datos leídos desde la lectora.

Parámetros opcionales

Para cada mandato, el usuario puede especificar unos parámetros suplementarios. Estos parámetros deben ir entre corchetes "[ ]" a continuación del nombre de fichero o del periférico "fuente". Algunos van seguidos de un valor decimal opcional.

B: Transferencia en modo bloque a bloque: los datos van formando "bufferes" hasta que se recibe el carácter XOFF <ctrl-S> emitido por el "fuente" para delimitar el tamaño de los bloques. PIP detiene la recepción y vacía los datos grabados en memoria sobre el fichero destino, después vuelve a leer los datos del dispositivo de entrada. Este parámetro se utiliza para transferir a disco los datos de un soporte secuencial (casete, por ejemplo).

Dn: Destruye los caracteres que pasan de la columna "n". Este truncado de línea se utiliza para copiar ficheros sobre soportes de papel que tengan un número limitado de caracteres por línea.

E: Modalidad "eco" en la consola para todos los datos transmitidos.

F: Anula los caracteres "form-feed o 0CH" (saltos de página).

H: Transfiere un fichero tipo HEX (formato Intel), con control de paridad longitudinal sobre una línea ("checksum").

I: Ignora los registros que empiezan por ":00" en un fichero con formato Intel o de tipo HEX.

L: Convierte las letras mayúsculas en minúsculas.

N: Añade un número de línea a cada línea transferida empezando en 1 y con un incremento de 1 en 1.

O: Transfiere un fichero objeto (no ASCII).

Pn: Inserta saltos de página cada n líneas. Si n=1 o no se especifica, el salto se efectúa cada 60 líneas.

Qs↑Z: Detiene la copia en el momento que se encuentra la cadena <s> seguida de <ctrl-Z>.

Ss↑Z: Rearranca la copia en el momento que se encuentra la cadena <s> seguida de <crtl-Z>.

Tn: Extiende los tabuladores para cada n-sima columna, durante la transferencia.

U: Convierte las letras minúsculas en mayúsculas.

V: Verifica que los datos se han copiado correctamente, realizando una lectura comparativa después de la escritura.

Z: Fuerza a cero el bit de paridad para cada carácter ASCII transferido.

Mejoras de la versión CP/M 2.2

Gn: Lectura de un fichero que pertenece al usuario "n".

W: Fuerza la escritura en ficheros R/O, protegidos contra escritura

R: Lectura del fichero de sistema (indicador SYS posicionado).

1.5.2.3. ED

Generalidades sobre el funcionamiento de ED

El editor de texto ED es un editor de líneas, al contrario de los editores de página o de pantalla. ED trabaja dentro del contexto del CP/M y permite construir y modificar ficheros de tipo texto codificados en ASCII.

Estos ficheros fuente están organizados como una serie de caracteres ASCII, separados por unos caracteres de fin de elemento (secuencia de <CR>, <LF> : carriage-return, line-feed). El tamaño de un elemento puede ser cualquiera y no está limitado siendo igual al número de caracteres tecleados antes del retorno de carro <CR>.

El editor ED dispone de un conjunto completo de mandatos definidos por una única letra, para crear y corregir los ficheros de texto.

Aunque el tamaño de memoria sea insuficiente para algunos pequeños sistemas gestionados por el CP/M, ED tiene la posibilidad de tratar ficheros de cualquier tamaño dentro de los límites permitidos por la configuración física. Si el fichero que va a editar es verdaderamente voluminoso y no cabe en memoria, el usuario puede llamar, por separado, a partes del fichero que se pueden introducir en el área de memoria reservada para la edición.

La llamada al editor se hace de la siguiente forma:

A><u>ED<nombre del fichero></u>

El nombre del fichero dado como argumento es un nombre explícito de la forma:

d: fichero.tip

Si el fichero no existe, se crea uno en vacío para poder escribir en él. Se visualiza entonces el mensaje NEW FILE seguido del caácter "*" que es el "prompt" del D, invitando al usuario a que teclee un mandato del editor. En principio, este mandato debe ser "I" (Insert) que hace que el editor se ponga en la modalidad de inserción para que el usuario pueda introducir su texto. El paso de la modalidad de inserción a la modalidad de mandato del ED se hace pulsando la tecla "Control-Z" del teclado. Aparece de nuevo el carácter de invitación "*" y ED se queda a la espera de un nuevo mandato. Se sale del editor con el mandato "E" (End). Se crea un duplicado del fichero fuente sobre otro fichero tipo BAK. Este fichero constituye una copia de seguridad del último fuente anterior a la edición. La utilidad del "Backup" es indispensable para evitar cualquier manipulación errónea; la versión precedente del fichero se conserva siempre en este fichero.

Ejemplo:

```
A>ED A:PRUEBA.TEX
  NEW FILE
     :  *I
  1  :  ESTA ES LA PRIMERA LINEA DEL FICHERO
  2  :  SE PASA A LA SEGUNDA LINEA PULSANDO <CR>
  3  :  TERMINAREMOS LA INSERCION EN LA SIGUIENTE LINEA
  4  :  <crt1-Z>
     :  *E      (salida del ED)
  A>
```

En el caso reflejado en la figura, en el que el usuario solicita al editor un fichero que ya existe, éste es el caso más frecuente, el fichero especificado queda abierto. El usuario puede entonces traer a memoria una parte o la totalidad del fichero mediante el mandato "A" (append), para listarle o modificarle, según desee. Al final del tratamiento, el área de memoria llamada "buffer de memoria" se vacía sobre un pseudo-fichero temporal de tipo "$$$", seguida del resto, no leído, del fichero fuente. Al fichero fuente se le cambia el nombre y se le da el tipo BAK y a su vez el fichero temporal "fichero.$$$" queda renombrado con su nombre original "fichero.tip".

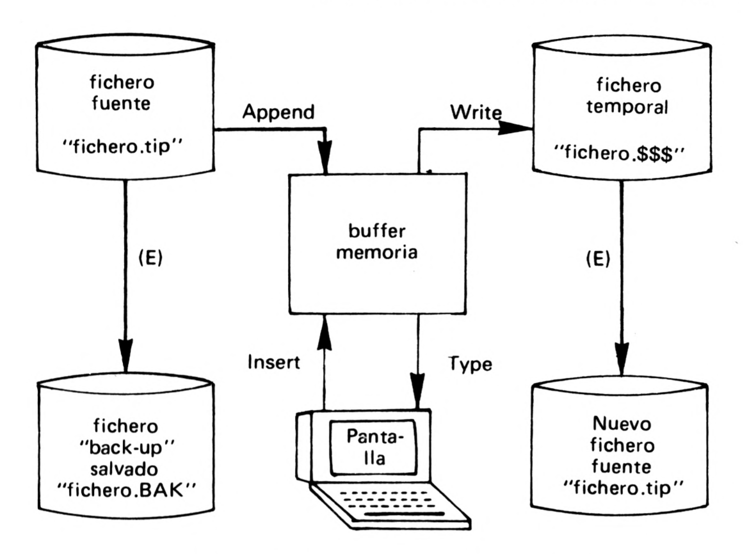


Fig. 8.— Ficheros manipulados por ED.

Organización del buffer de memoria

El buffer de memoria es un área reservada en la cual se coloca el texto fuente una vez que el usuario ha introducido los mandatos "A" (Append) o "I" (Insert". Este buffer está asociado al apuntador de carácter actual "CP", también llamado "cursor". Con cada introducción de una nueva línea el buffer se hace más pequeño. Es decir, este buffer constituye un flujo continuo de caracteres.

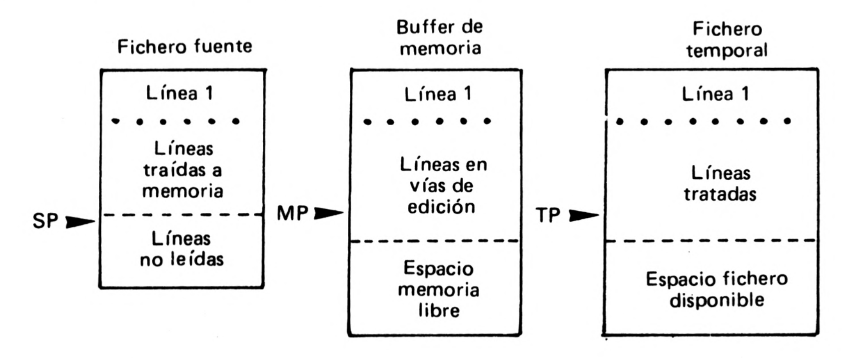


Fig. 9.— Aspecto del buffer de memoria en relación con los ficheros.

Los tres apuntadores SP (apuntador del fuente, "source pointer"), MP (apuntador de memoria, "memory pointer") y TP (apuntador del fichero temporal, "temporary pointer") se actualizan en cada transferencia de texto (mandatos: A, W, E, H, O, R).

El apuntador CP (apuntador de carácter actual, "character pointer"), se actualiza con cada operación que se realice sobre el texto en el buffer de memoria (listado, modificación, posicionamiento, etc...). CP varía entre el principio y el final del buffer (MP) y está asociado a la línea actual CL (current line) del texto.

Operaciones de transferencia de texto

nA (append) Trae las "n" primeras líneas no tratadas del fichero fuente (localizadas por SP), a continuación del buffer de memoria (localizado por MP). Los apuntadores SP y MP se incrementan en "n".

nW (write) Escribe las "n" primeras líneas del buffer de memoria en el fichero de memoria a partir de la posición indicada por TP. Se libera el espacio correspondiente de memoria y las líneas siguientes se desplazan hacia el comienzo del buffer, con lo que se provoca un reagrupamiento de la memoria. El apuntador MP se decrementa en "n" mientras que el apuntador TP se incrementa en "n".

+/–U (upper) Convierte las minúsculas en mayúsculas si "+U", si no, no se realiza ninguna conversión.

El valor de "n" puede variar de 1 (valor por defecto que puede omitirse) a 65535, también simbolizado por el carácter "#". Así pues, el mandato "#A" se utiliza normalmente para traer a memoria todo el fichero fuente. Si el fichero fuente es demasiado voluminoso, el usuario puede llamar al fichero trozo a trozo, utilizando el mandato "0A" que rellena la mitad del buffer de memoria.

Operaciones de transferencia de fichero

E (end) Salida normal del editor. El buffer de memoria se vuelve a copiar en el fichero temporal, así como el resto de las líneas no leídas del fichero fuente. El antiguo fichero fuente queda renombrado con el tipo BAK y el fichero temporal $$$ con el nombre del fichero fuente. A continuación se devuelve el control al CCP.

H (head) Idéntico al mandato E pero se queda a la espera de un nuevo mandato del editor para el mismo fichero fuente.

O (original) Se abandona la edición en curso y se vuelve a llamar al fichero fuente que todavía está sin modificar. Todos los apuntadores se ponen de nuevo a cero.

Q (quit) Se abandona la sesión de trabajo sin alterar ningún fichero y se devuelve el control al monitor.

Operaciones sobre el buffer de memoria

Toda operación realizada en el buffer de memoria pone en juego el apuntador de carácter actual CP, así como el de la línea actual CL.

I (insert) Inserta una o varias líneas delante de la línea actual CL. Todas las líneas tecleadas deben terminar con un retorno de carro <CR>; el carácter <LF> "line-feed" se añade automáticamente. La parada de la inserción se materializa mediante el tecleo del carácter <ctrl-Z>, con lo que el editor envía el prompt "*".

+B (begin) Posiciona el cursor CP al comienzo del buffer de memoria.

–B (bottom) Posiciona el cursor CP al final del buffer de memoria.

+/–nC (cursor) Desplaza el cursor CP "n" caracteres hacia delante si "+", o "n" caracteres hacia atrás si "–". Los caracteres de fin de línea <CR> y <LF> se contabilizan igual que los otros caracteres.

+/–nD (delete) Suprime los "n" caracteres situados detrás del apuntador CP si "+"; si "–", suprime los "n" caracteres situados delante.

+/–nK (kill) Suprime "n" líneas del texto fuente. Posteriores si "+" y anteriores si "n", relativas al apuntador CP.

+/–nL (line) Posiciona el cursor al comienzo de la n-sima línea relativa a la línea actual CL. Si n=0 el posicionamiento se efectúa sobre la línea actual. Si el valor de "n" se

sale de los límites del buffer de memoria, el cursor se posicionará al comienzo o al final del buffer.

+/–nT (type) — Visualiza el contenido de las (n–1) líneas relativas a la posición del cursor CP. Si n = 0 se visualizan los caracteres desde el comienzo de la línea actual hasta la posición del cursor. Si n=1 (valor por defecto), se visualizan los caracteres existentes entre el cursor y el final de la línea.

+/–n — Es equivalente al mandato "+/–nLT". Si se omite n, se lista la línea siguiente (signo +), si no, se lista la línea anterior (signo –).

+/nP (page) — Imprime "n" secuencias de 24 líneas (página de pantalla) en relación con la línea actual. P sola, imprime la página siguiente, –P imprime la página anterior y 0P imprime la página actual.

Búsqueda y modificación de cadenas

nF <cadena> (find) — Búsqueda de la cadena de caracteres <cadena> en el buffer de memoria, a partir de la posición actual del cursor CP. Si se ha especificado "n", el posicionamiento se realizará sobre la n-sima cadena encontrada. Si la búsqueda es infructuosa el CP se mantendrá en su posición inicial y se enviará al terminal el mensaje "BREAK at #".

I<cadena> (insert) — Inserta, a continuación del carácter apuntado por CP, la cadena de caracteres especificada.

nN<cadena> (n-sima) — Búsqueda de la "n-sima" ocurrencia de la cadena especificada no en el buffer de memoria sino directamente en el fichero fuente. En el caso de que el buffer de memoria se sature durante la búsqueda, se efectúa una copia automática de éste sobre el fichero temporal (mandato #W). Una vez que se ha encontrado la cadena, se rellena el buffer de memoria hasta la mitad con las líneas siguientes del fichero fuente.

nS<cadena 1> ↑ Z — Realiza la sustitución de la cadena n-sima,

<cadena 2> (substitute)	<cadena 1> por la cadena <cadena 2>.
nJ<cadena 1> ↑ Z <cadena 2> ↑ Z <cadena 3> (yuxtaposición)	Yuxtapone la cadena <cadena 2> a la cadena <cadena 1> y destruye todos los caracteres hasta encontrar la cadena <cadena 3>. Si se omite <cadena 3>, no se realiza ninguna supresión, es un caso de simple yuxtaposición.

Operaciones anexas

Sobre la misma línea se puede teclear un número cualquiera de mandatos. No se ejecutarán hasta que se detecte el retorno de carro. Si el usuario comete un error de tecleo tiene la posibilidad de utilizar los caracteres especiales de edición del teclado:

<rubout> : borra el último carácter tecleado;
<ctr l-U> : borra toda la línea en curso;
<ctr l-E> : borra toda la línea y envía un retorno de carro;
<ctr l-C> : re-inicializa el sistema CP/M (idéntico a "Quit").

Los mandatos que se pueden combinar son los que operan sobre el buffer de memoria (mandatos: B, C, D, K, L, T y P).

Ejemplo de cadena de mandatos:

:*B3L-T posicionarse en la línea 3 (3L) en relación al comienzo (B) e impresión de la línea (–T).

También se dispone de un "macro-mandato" para ejecutar de manera repetitiva una secuencia de mandatos del ED; se trata del mandato "M", cuya sintaxis es:

nM<cadena de mandatos>

Se ejecuta la cadena de mandatos "n" veces si n>1, si no, se ejecuta en forma repetitiva hasta que se presenta una condición anormal (final de buffer, por ejemplo).

Mensajes de error

Durante la fase de edición (mandatos escritos desde el teclado), el editor de textos ED visualiza el último carácter leído seguido de un código de error simbolizado por uno de los caracteres siguientes:

? mandato inexistente;
> buffer de memoria saturado (para los mandatos D, K, N y S) o cadenas demasiado largas (para los mandatos F, N y S);
no satisface todas las condiciones ligadas al mandato;
0 fichero biblioteca de tipo LIB no encontrado.

Pueden surgir otros errores que no están ligados directamente a la edición del texto. En particular, si se detecta un error físico sobre un disco se visualiza el siguiente mensaje:

PERM ERR DISK d

donde "d" es el nombre del disco actualmente seleccionado. El usuario tiene entonces la opción de ignorar el error pulsando cualquier tecla o de reinicializar el sistema mediante ↑C.

Mejoras de la versión CP/M 2.2

El acceso a los números de líneas, en forma relativa, es decir, en relación a otro número de línea, es poco flexible y muy pesado. La versión CP/M 2.2 ofrece la posibilidad de acceder a los números absolutos de las líneas. En este caso, el carácter ":" debe ir a continuación del número de línea:

:*10:T lista la línea número 10.

Igualmente, el usuario puede hacer una referencia absoluta a una línea, a partir de la posición actual del cursor; en este caso el número absoluto debe ir precedido del carácter ":"

:*:30T lista las líneas desde la línea actual hasta la línea 30
:*15::20K suprime desde la línea 15 hasta la línea 20.

1.5.2.4. ASM

El ensamblador traduce un fichero fuente (source) escrito en el lenguaje simbóligo del 8080 y produce un fichero objeto (objet) en for-

mato Intel 8080 en hexadecimal. La llamada al ensamblador ASM acepta dos tipos de sintaxis.

ASM <nombre de fichero>

o

ASM <nombre de fichero> . parámetros

El nombre de fichero está implícito (de tipo ASM) y no debe ir seguido de su tipo ASM. Los parámetros opcionales definen las localizaciones (números de los discos) de los ficheros involucrados en el ensamblaje. Estos ficheros son tres:

el fichero fuente	"fichero.ASM"
el fichero objeto	"fichero.HEX"
el fichero listado	"fichero.PRN"

La forma completa de la sintaxis de llamada es:

ASM<nombre de fichero >. <s><h><p>

los parámetros <s>, <h> y <p> representan respectivamente los nombres de los discos (sin los :) del fichero fuente (<s>), del fichero objeto (<h>) y del fichero imagen para la impresora (<p>). Estos nombres pueden ser:

– los símbolos A a P (del disco 0 al disco 15).
– el símbolo X para <p> (listado sobre pantalla)
– el símbolo Z para <h> y <p> (inhibición del fichero de salida).

Ejemplo:

A>ASM PROGR.BAB

va a buscar el fichero fuente PROGR.ASM al disco B:, produce un fichero objeto PROGR.HEX en el disco A:. como resultado del ensamblaje y genera un fichero imagen de la impresora PROGR.PRN en el disco B:

Asímismo:

A>ASM PROGRAMM.AZX

ensambla el fichero PROGRAMM.ASM del discoA: produciendo un listado resultado en el terminal sin construir el fichero objeto.

Formato de un programa ensamblador

Como en la mayoría de los lenguajes de ensamblaje, cada línea puede descomponerse en cuatro partes denominadas "campos".

etiqueta operación operando ;comentario

Toda línea en lenguaje ensamblador debe terminar con la secuencia <CR> <LF>. El carácter "!" juega también un papel de separador de línea en ensamblador, permitiendo la escritura de varias instrucciones en la misma línea física.

– El campo etiqueta no es obligatorio; sirve como referencia simbólica para las variables o para dar un nombre a una dirección de bifurcación dentro del programa (indicativos, bucles, etc...). Caso de existir, empezará en la columna 1 estando formada por una cadena de hasta 16 caracteres alfanuméricos. El primer carácter no puede ser un dígito. Por razones de compatibilidad con otros ensambladores 8080, el campo etiqueta puede terminar con el carácter ":".

Etiquetas correctas	Etiquetas incorrectas
VARIABLE ETIQ 18: bucle: LABEL$ 1982 symbolomuyl:	"VARIABLE" 18ETIQ: (bucle): LABEL$ 1982 símbolo muylar:

– El campo de operación contiene una directiva de ensamblaje o el mnenónico de una instrucción en lenguaje máquina 8080. La lista de los códigos de instrucción se da en uno de los anexos.

– El campo operando está reservado a la parte variable de la instrucción. Permite definir una dirección, un valor numérico, una referencia simbólica (símbolo) o incluso cualquier expresión calculable. Algunas instrucciones simples no utilizan este campo.

– El campo comentario es opcional e ignorado por el ensamblador. Empieza con el carácter ";" y termina con el final de la línea física. Se pueden introducir los comentarios en cualquier lugar del texto fuente. Por razones de compatibilidad, si aparece el carácter "*" en la primera columna de una línea, indica que toda la línea es de comentario.

Palabras reservadas

Algunos símbolos no pueden ser redefinidos por el programador, porque están reservados para referenciar los registros del micro-ordenador 8080 o Z80

A	registro A
B	registro B
C	registro C
D	registro D
E	registro E
H	registro H
L	registro L
M	contenido del par de registros HL
SP	registro apuntador de pila (Stack pointer)
PSW	registro palabra de estado del programa (program status word)

Operadores aritméticos y lógicos

Los operandos pueden estar formados por expresiones que combinen otros operandos mediante operaciones aritméticas o lógicas. Los operadores reconocidos son:

+	suma aritmética
–	diferencia aritmética o menos unario
*	multiplicación entera sin signo
/	división entera sin signo
MOD	resto de la división
NOT	inverso lógico o complemento a uno
AND	intersección lógica
OR	unión lógica
XOR	O exclusiva
SHL n	desplazamiento a la izquierda de "n" bits rellenando con ceros
SHR n	desplazamiento a la derecha de "n" bits rellenando con ceros.

En el caso de que existan varios operadores se recomienda la utilización de paréntesis. De hecho, el cálculo de una expresión se realiza de

izquierda a derecha aplicándose a algunos operadores una serie de reglas llamada prioridad de los operadores. Los operadores con igual prioridad se evalúan de izquierda a derecha, según se van encontrando dentro de la expresión. El orden jerárquico de prioridades se define de la siguiente forma:

*/ MOD SHL SHR (prioridad más alta)
– +
NOT
AND
OR XOR (prioridad más baja)

Directivas de ensamblaje

Las directivas de ensamblaje aparecen, bajo la forma de mnemónicos, en el campo de operación pero no son ejecutables en el sentido de las instrucciones. A continuación se detallan las directivas reconocidas por el ensamblador ASM:

ORG Define la dirección de implantación de la totalidad o de una parte del programa que sigue a esa directiva. La sintaxis general de la directiva ORG es:

etiqueta ORG expresión ;comentario

Sólo es obligatorio el campo expresión. Pueden existir varias directivas ORG en el seno de un mismo programa. Un programa bajo CP/M empieza generalmente con ORG 100H.

END Indica al emsamblador el final del programa a ensamblar.

etiqueta END expresión ;comentario

Si se omite la expresión, la dirección de comienzo de la ejecución que se toma por defecto es 0000H. En caso contrario, la expresión, una vez evaluada, define la dirección de partida del programa. Para un programa CP/M estándar tendremos: END 100H

EQU Permite establecer una equivalencia o un sinónimo entre la expresión escrita a la derecha y el símbolo del campo etiqueta. La sintaxis es:

```
etiqueta      EQU      expresión      ;comentario
```

La utilización de la directiva EQU permite al programador clarificar su programa, hacerle mucho más cómodo de leer mediante la elección de símbolos personalizados y construirle de forma que sea fácilmente parametrizable.

SET — Esta directiva es idéntica a la directiva EQU con la excepción de que el símbolo del campo etiqueta puede volver a definirse en distintos puntos a lo largo del programa.

IF,ENDIF — Estas dos directivas permiten que el ensamblaje sea condicional. Su sintaxis es:

```
IF    expresión
         ...
secuencia de instrucciones
         ...
ENDIF
```

Se calcula la expresión que va a continuación del IF y si su valor es distinto de cero (condición verdadera) se ensambla la siguiente secuencia de instrucciones; si el valor es cero (condición falsa) el ensamblador salta la secuencia que hay entre IF y ENDIF y reanuda el ensamblaje después del ENDIF.

DB, DW, DS — Estas directivas permiten inicializar y reservar áreas de memoria.

- DB Inicializa un área de memoria octeto a octeto (8 bits)
- DW Inicializa un área de memoria palabra a palabra (16 bits)
- DS Reserva un área de memoria del tamaño especificado en octetos.

Las sintaxis posibles son:

```
etiqueta   DB    expr1,expr2,expr3,...    ;comentario
etiqueta   DW    expr1,expr2,expr3,...    ;comentario
etiqueta   DS    expresión                ;comentario
```

Las expresiones expr1, expr2, expr3, etc., pueden ser contantes numéricas, constantes de tipo cadenas de caracteres (textos) o incluso símbolos. La directiva DS sólo ad-

mite una expresión que represente el tamaño en octetos del área a reservar. La directiva DW genera el valor de las expresiones, ajustado a la derecha, en palabras de 16 bits, con el octeto de menor peso colocado en primer lugar y el octeto de mayor peso en segundo lugar.

Mensajes de error

Se distinguen dos categorías de diagnósticos de errores: los errores debidos al propio ensamblador y los errores debidos al ensamblaje del programa fuente. En la primera categoría podemos señalar los mensajes de errores siguientes:

NO SOURCE FILE PRESENT	No existe el nombre del fichero especificado a continuación del mandato ASM. Verificar si se ha indicado el disco correcto o si se ha cometido algún error de tecleo.
NO DIRECTORY SPACE	El directorio del disco está saturado. Es aconsejable destruir aquellos ficheros que no sean precisos en ese momento (ficheros tipo BAK o PRN).
SOURCE FILE NAME ERROR	El nombre del fichero a ensamblar debe ser explícito. No se pueden utilizar los símbolos "?" y "*"
SOURCE FILE	Se ha detectado un error físico en un sector del disco del fichero fuente.
OUTPUT FILE	No queda espacio disponible en el disco o el disco está protegido. Obrar en consecuencia: cambiar o desproteger el disco.
CANNOT CLOSE FILE	El disco está protegido contra escritura.

En la segunda categoría tenemos los errores a nivel del programa fuente. Podemos encontrar los siguientes tipos de errores:

D: No es válido el valor de la expresión (demasiado largo).
E: No es válida la expresión o su valor es demasiado grande.
L: El campo etiqueta es incorrecto.
N: Instrucción no implementada en ASM (ver MAC-80).
O: Overflow: no se puede tratar la expresión por ser demasiado complicada.

P: El valor de la etiqueta ha sido modificado en el curso del programa.
R: El registro especificado no es compatible con los mnemónicos.
U: El símbolo del campo operando no está definido en ninguna parte.
V: La expresión del campo operando es incorrecta.

1.5.2.5. LOAD

El mandato LOAD produce un fichero ejecutable de tipo COM a partir de un fichero objeto como resultado de un ensamblaje de tipo HEX en formato Intel 8080 (binario codificado en hexadecimal). Como argumento del mandato LOAD se escribe el fichero tipo HEX, sin dar su tipo ya que se considera implícito. La sintaxis, de hecho, es muy sencilla.

LOAD <nombre del fichero>

LOAD edita una secuencia de informaciones de la siguiente forma:

FIRST	ADDRESS	0100	0100= dirección de comienzo del programa
LAST	ADDRESS	aaaa	aaaa= dirección de final del programa
BYTES	READ	nnnn	nnnn= número de octetos leídos
RECORDS	WRITTEN	rr	rr= número de registros leídos

Si el fichero especificado no es del tipo HEX se envía un mensaje al usuario. Si la operación se ejecuta correctamente se crea un fichero ejecutable de tipo COM y con el mismo nombre, conservándose el fichero tipo HEX.

El fichero tipo COM puede entonces ejecutarse directamente en la TPA, tecleando sencillamente el nombre (sin añadir el tipo que está implícito). Es el módulo CCP del CP/M el que contiene el cargador para implantar en el área TPA los programas de tipo COM.

1.5.2.6. DDT

El programa DDT (Dynamic Debugging Tool) es una herramienta indispensable para la puesta a punto y comprobación de los programas desarrollados en un entorno CP/M. La llamada a este mandato puede ser de dos formas:

DDT

o

DDT d: nombre de fichero.tipo

donde d: es el nombre del disco en el que está el fichero especificado. El tipo puede ser cualquiera si de lo que se trata es de modificar el fichero en cuestión, si no, el tipo es COM que es el más normalmente utilizado cuando se trata de una ejecución controlada por DDT. Si se especifica el tipo HEX, DDT convierte el formato Intel en binario, igual que hacía el mandato LOAD.

La primera forma sintáctica es equivalente a la segunda si se da la siguiente secuencia de mandatos:

A>DDT	llamada al DDT
–id: nombre del fichero.tipo	apertura del fichero en entrada
–R	carga del fichero en memoria

Se envía un mensaje de bienvenida a la consola:

nnK DDT VER v.v

donde "nn" es el tamaño de memoria en Kilo-octetos y "v.v" el número de versión. A continuación DDT envía:

NEXT PC
nnnn pppp

donde "nnnn" es la primera dirección de comienzo de ejecución del programa (contador ordinal).

DDT emite su propio "prompt": el carácter "–" invitando al usuario a que teclee un mandato interno del DDT. En cualquier momento el usuario puede salir del DDT tecleando <ctrl-C> o G0 (bifurcación a la dirección 0000H). La imagen en memoria de su programa se puede salvar mediante el mandato:

SAVE nn "nombre del fichero.COM"

donde "nn" representa el número de páginas de 256 octetos del programa. El fichero salvado de esta manera, puede ser ejecutado de nuevo bien directamente, tecleando el nombre del fichero después del prompt A >, bien a través del DDT para una nueva puesta a punto.

Los mandatos del DDT

Antes de teclear ningún mandato interno del DDT, el usuario debe esperar a que aparezca el carácter de invitación "–" indicando que el DDT está ya listo. Los mandatos del DDT se formulan con una sola letra con o sin argumentos (valor numérico o dirección).

Estos mandatos permiten ensamblar instrucciones del 8080, desensamblar del binario, visualizar y modificar áreas de memoria y seguir o pararse en cualquier instrucción. El conjunto de estas funciones se detalla a continuación:

A (Assemble) Aa a = dirección hexadecimal

Este mandato permite ensamblar instrucciones 8080 "sobre la marcha". A diferencia del ASM, no hay campo de etiqueta y por lo tanto no existe gestión de símbolos. Todas las referencias de direcciones están representadas por su valor absoluto y en hexadecimal. Una línea de ensamblaje se termina con un <cr> visualizándose entonces la dirección siguiente para aceptar una nueva línea de ensamblador. El ensamblaje finaliza cuando se teclea una línea vacía, es decir, sólo un <cr>.

Ejemplo:

```
–A100
100 JMP 31A
103 <cr>
```

D (Display)

D

Di i = dirección inicial

Di,f f = dirección final

En el primer caso, se visualiza, sobre doce líneas de la consola, el contenido de los 192 octetos, a partir de la dirección actual. Cada línea contiene dieciséis octetos en hexadecimal, está precedida de una dirección hexadecimal y seguida de la traducción ASCII de dichos octetos. Los octetos no imprimibles son sustituidos por un ".".

F (Fill) Fi,f,v, v = valor del octeto

El mandato F escribe el valor hexadecimal "v" en las posiciones de memoria comprendidas entre la dirección inicial y final.

Ejemplo: F1800,1A7F,FF

G (Go)

G
Gd d = dirección de arranque
Gd,p p = dirección de parada
Gd,p,b b = segunda dirección de parada
Gp p = dirección de parada
Gp,b p,b = direcciones de parada

En el primer caso la ejecución del programa comienza en la dirección actual del contador de instrucciones, 100H por defecto. En el segundo caso, la dirección de comienzo de la ejecución es la especificada. El tercer caso fuerza una parada en la dirección indicada (breakpoint) cuando el contador de instrucciones alcance es ese valor. El cuarto caso permite dos puntos de parada. En los dos últimos casos, la dirección de arranque es la dirección actual, es decir, la dirección en la que el usuario se hubiera parado anteriormente, por ejemplo. Los puntos de parada se borran automáticamente cuando el programa ya ha pasado por ellos.

Ejemplo: G100,158

H (Hexa)

Ha,b a,b = valores hexadecimales

Este mandato permite realizar la suma y la diferencia de dos valores numéricos "a" y "b" codificados en hexadecimal.

Ejemplo:
```
:–H1580,724
1CA4 0E5C
```

I (Input)

Id: nombre del fichero.tipo

Este mandato permite la apertura, para lectura, de un fichero, colocando su FCB en la posición 5CH de la memoria (FCB implícito).

Ejemplo: –IB:TEST.COM

L (List)

L
Li i = dirección inicial
Li,f f = dirección final

Este mandato permite desensamblar el código máquina 8080 en ensamblador simbólico 8080. Si L no lleva ningún argumento, se desensamblan doce instrucciones

a partir de la dirección actual, lo cual representa la mitad de una pantalla. Dos mandatos L sucesivos llenan por completo la pantalla del terminal.
También se puede especificar una dirección de comienzo y otra de final.

Ejemplo:

```
-L100,108
100 LXI D,128
103 DAD D
104 SHLD 0122
107 MVI A,0
108 MOV B,E
109
```

M (Move)

Mi,f,a   a = dirección destino
Este mandato permite transferir el área de memoria comprendida entre las direcciones inicial y final a otro área de memoria definida por la dirección destino. La dirección inicial se incrementa en 1 con cada transferencia de un octeto y si se sobrepasa la dirección final se detiene la transferencia.

Ejemplo: -M1200,127F,0080

R (Read)

R
Rd   d = desplazamiento
Este parámetro permite cargar en memoria el fichero descrito en el FCB implícito en la posición 005CH de la memoria. Los ficheros a cargar en memoria son del tipo HEX o COM. Si se especifica el argumento "d", se añadirá este valor a cada dirección del programa a cargar. El mandato no tendrá efecto si antes no se ha ejecutado el mandato I.

S (Set)

Sd   d = dirección de memoria
Este mandato permite examinar, y opcionalmente modificar, el contenido de la posición de memoria especificada. DDT devuelve la posición de memoria seguida de su contenido. Si el usuario teclea un <cr> no se altera el contenido y se visualizan la dirección siguiente y su contenido. La modificación de un octeto de memoria se realiza tecleando, al lado del contenido actual, un

valor hexadecimal. Esta operación finaliza con el tecleo del carácter "." que permite el retorno al nivel de mandatos del DDT.

Ejemplo:
```
-S103
103 4D<cr>
104 34 1A<cr>
105 18 .<cr>
```

T (Trace) T
Tn n = número de pasos
Este mandato permite un seguimiento selectivo del programa que se está ejecutando. El argumento "n" determina el número de instrucciones a seguir; si se omite, sólo se hace el seguimiento de una instrucción. El formato de una instrucción afectada por este mandato es el siguiente:

```
CfZfMfEfIf A=aa B=bbcc D=ddee H=hhll S=ssss P=pppp instruction
```

donde "f" es el valor (0 ó 1) del flag asociado:

C flag Carry
Z flag Zero
M flag Minus
E flag Parité paire
I flag Carry Interdigit.

A registro acumulador (octeto de ocho bits)
B par de registros: B (pesos fuertes), CP/M (pesos débiles)
D par de registros: D (pesos fuertes), E (pesos débiles)
H par de registros: H (pesos fuertes), L (pesos débiles)
S registro apuntador de la pila (16 bits)
P contador de la instrucción en curso(16 bits); instrucción: mnemónico de la instrucción en curso.

En el caso en el que el seguimiento (trace) se detuviera, la dirección de la siguiente instrucción a ejecutar aparece al final de la última línea, después de la instrucción desensamblada, con el formato: * <dir>.
El seguimiento se puede interrumpir en cualquier mo-

mento pulsando la tecla <rubout>; el control se devuelve al DDT.

U (Untrace) Idéntico al mandato T
Este mandato es el mismo que el mandato T con la excepción de que no siguen las etapas intermedias del programa que se está ejecutando. En la consola sólo aparece la última línea del seguimiento. Este mandato es una variante de la combinación del mandato G con un punto de parada y del mandato X.

X (Examine) X
Xr registro = registro de la CPU
Este parámetro permite visualizar el contenido de todos los registros de la CPU, si se especifica con el primer formato, siendo la visualización la misma que la del mandato T.
En el segundo caso se especifica el nombre del registro o del flag visualizándose su contenido en hexadecimal en 1, 8 ó 16 bits dependiendo de si se trata de un flag, del acumulador o de un par de registros. Este contenido se puede modificar escribiendo un nuevo valor al lado del contenido antiguo.

Funciones del DDT

Cuando un programa de usuario está siendo controlado por el mandato DDT, el área TPA está ocupada por dos programas: el programa del usuario y el programa DDT. En el mandato de la llamada al DDT, éste se carga como un programa estándar en el área TPA, en la dirección 100H; después se "autotraslada" de manera que el final del DDT sea contiguo a la dirección base del BDOS. De esta forma queda liberado el espacio de memoria del usuario para introducir el programa que se desea comprobar. La dirección del punto de entrada de los servicios del BDOS, en la 0006H de la memoria, queda sustituida por la dirección de comienzo del núcleo del DDT que representa, en ese momento, la dirección del final lógico de la memoria utilizable por el programa.

El DDT posee una estructura de solapamiento que permite que algunos programas voluminosos se "sobre-escriban" en la zona que no es vital del DDT. De hecho, el DDT consta de dos partes: el núcleo indispensable para la puesta a punto y el módulo ensamblador-desensamblador que es el que se puede solapar con el final de un programa

en la TPA. Si se produce dicho solapamiento ya no se pueden utilizar los mandatos A y L; los mandatos del seguimiento (trace) dan el contenido de la instrucción en hexadecimal.

Los mandatos que afectan al contador de instrucciones (o de programa) P (G, T y U) utilizan la instrucción de interrupción lógica RST 7 y su situación en el área TPA de la memoria.

1.5.2.7. SUBMIT

SUBMIT es un programa de utilidad que permite ejecutar una secuencia de mandatos CP/M. El usuario crea un fichero de mandatos con la ayuda del editor de texto evitando así el tener que teclear cada vez sus mandatos por separado.

Opcionalmente se pueden utilizar parámetros formales para que el usuario pueda introducir sus propios parámetros a nivel de la ejecución del mandato.

La sintaxis general del mandato es:

SUBMIT <nombre del fichero> <parámetros>

El fichero referenciado debe ser de tipo SUB aunque no aparece en el nombre del fichero dado como argumento del mandato. El campo parámetros es facultativo y puede estar constituido por uno o varios parámetros escritos en la misma línea. Cada uno de estos parámetros puede referenciar un nombre de fichero o una información necesaria para el fichero de mandato SUB.

A estos parámetros especificados en el mandato escrito desde el teclado, están asociados, en el fichero de mandato, los símbolos formales:

$1 $2 $3 ... $n

Cada símbolo formal numerado de esta manera, referencia al parámetro situado dentro de la lista, en la posición correspondiente; el primer parámetro está asociado a $1, el segundo a $2, etc...

Sea el fichero de mandatos JOB.SUB construido con el editor de texto ED:

```
ASM $1
LOAD $1
ERA $1.HEX
PIP $2=$1.PRN
ERA $1.PRN
$1
```

El mandato:

A>SUBMIT JOB PROGRAM CON:

efectuará el ensamblaje del fichero PROGRAM.ASM, construirá el fichero ejecutable PROGRAM.COM, destruirá el fichero objeto intermedio PROGRAM.HEX, listará por la consola el fichero de impresión PROGRAM.PRN, destruirá este fichero y, por último, cargará y ejecutará el fichero PROGRAM.COM.

SUBMIT crea un fichero intermedio $$$.SUB en el que se realizan todas las sustituciones de los parámetros formales $i por los parámetros reales introducidos desde el teclado. Cada línea de este fichero representa un mandato. SUBMIT ejecuta secuencialmente estos mandatos hasta que se agota el fichero.

1.5.2.8. DUMP

DUMP permite visualizar en la consola, en código hexadecimal, el contenido de un fichero cualquiera.

La visualización del fichero se hace línea a línea. Cada línea contiene una serie de doce octetos traducidos en hexadecimal y está precedida por una dirección de cuatro octetos relativa al comienzo del fichero (codificada en hexadecimal).

Se puede suspender el proceso de la visualización en cualquier momento si se está realizando demasiado rápidamente (en pantalla a 9600 baudios, por ejemplo), pulsando la tecla <ctrl-S> (XOFF) para detener la imagen. El proceso se desencadena de nuevo pulsando cualquier tecla. Asímismo, la interrupción definitiva de la visualización se realiza tecleando un carácter cualquiera (si el fichero es demasiado largo, por ejemplo).

Ejemplo de visualización con DUMP:

```
A>DUMP PROG.COM

0000   42 49 4F 53 22 20 3D 20 24 02 C3 87 0C CD 68 21
0010   00 00 39 22 DE 01 31 00 02 2A 06 00 22 48 01 23
0020   23 23 22 4A 01 2A 01 00 23 5E 23 56 1B 1B 1B EB
0030   22 4C 01 11 03 01 0E 09 CD 05 0D 2A 48 01 CD A9
```

1.5.2.9. MOVCPM

El programa MOVCPM permite reconfigurar el sistema CP/M para cualquier tamaño de memoria. Se pueden introducir dos parámetros

opcionales para indicar el tamaño deseado, en kilo-octetos, del nuevo sistema y la acción a tomar al final del tratamiento.

Si se omite el segundo parámetro (carácter "*"), MOVCPM reconstruye una nueva imagen de memoria del CP/M con el tamaño deseado (primer parámetro) y cede el control a este nuevo CP/M sin salvarlo en disco. Este es un método para probar una nueva configuración del CP/M que no entraña ningún peligro.

Por el contrario, si se especifica el segundo parámetro "*", la nueva imagen del CP/M representa una imagen en disco del CP/M lista para ser salvada en disco a través de los mandatos SAVE o SYSGEN. En este caso se visualiza el siguiente mensaje:

```
READY FOR "SYSGEN" o
"SAVE 34 CPMnn.COM"
```

donde "nn" representa el tamaño de la memoria en Kilo-octetos (primer argumento de MOVCPM).

La imagen de memoria situada entre las direcciones 0900H y 2300H constituye la imagen en disco del CP/M: el BOOT en la 900H, el CCP en la 980H, el BDOS en la 1180H y el BIOS en la 1F80H. Se puede salvar esta imagen de memoria con el mandato SAVE en el fichero "CPMnn.COM", el cual puede ser modificado por el mandato DDT para dar lugar a un nuevo sistema CP/M mediante el mandato SYSGEN.

Mandatos válidos:

MOVCPM	Construye una imagen del nuevo CP/M en memoria con un tamaño máximo y da control a este nuevo CP/M.
MOVCPM nn	Construye una imagen de un CP/M configurado, a "nn" K-octetos en memoria y le cede control.
MOVCPM **	Construye una imagen de disco en memoria de un CP/M configurado con un tamaño máximo para salvarlo posteriormente con SAVE o SYSGEN.
MOVCPM nn *	Construye una imagen de disco en memoria de un CP/M configurado con un tamaño de "nn" K-octetos con un tamaño de "nn" K-octetos para salvarlo posteriormente con SAVE o SYSGEN.

1.5.2.10. SYSGEN

El mandato SYSGEN permite generar un sistema CP/M sobre un nuevo disco. De hecho, SYSGEN copia el CP/M sobre otro disco. Esta operativa se realiza normalmente para crear disquetes CP/M "bootables" ya sea a partir del CP/M actual, de una imagen de memoria de un CP/M reconfigurado mediante MOVCPM o incluso a partir de un CP/M modificado en memoria con el mandato DDT.

SYSGEN construye pues, un sistema CP/M sobre el disquete de destino, que puede ser utilizado, a partir de ese momento, como disquete de sistema.

Este mandato dialoga con el usuario en forma interactiva con preguntas claras a fin de evitar errores en la manipulación.

Ejemplos del funcionamiento de SYSGEN:

```
A>SYSGEN
SYSGEN VER x.x
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
SOURCE ON A:,THEN TYPE RETURN <cr>
FONCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION ON B:,THEN TYPE RETURN <cr>
FONCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) <cr>
A>
```

1.6. SERVICIOS DE SISTEMA OFRECIDOS

Podemos considerar que el CP/M ofrece dos categorías de servicios de sistema: las funciones del BDOS y las primitivas del BIOS.

1. Las funciones lógicas ofrecidas por el BDOS son independientes de la configuración física y cualquier programador puede acceder a ellas. En efecto, el acceso a estas funciones se realiza por un paso obligado en el que se aplican unos controles estrictos sobre la validez y coherencia de los parámetros transmitidos.

2. Las primitivas de entrada-salida del BIOS están ligadas a la configuración física: consola, impresora, disquete o disco duro.

Estas primitivas deben ser tratadas con la mayor precaución y se desaconseja su utilización a los programadores poco expertos.

En lo que sigue de este capítulo analizaremos todos estos servicios.

1.6.1. Las funciones de sistema de BDOS

1.6.1.1. El punto de entrada de los servicios de sistema

El acceso a las funciones del CP/M se obtiene proporcionando el número de la función y opcionalmente una información suplementaria a través del punto de entrada situado en la dirección 0005H de la memoria, mediante la instrucción de llamada CALL.

El número de función pasa al registro "C" y la información suplementaria al par de registros "DE". La información devuelta después del "CALL" se sitúa en un octeto en el registro A o en una palabra sobre el par de registros "HL".

Ejemplo de llamada de un servicio monitor:

```
BDOS EQU 0005H           ; al comienzo del programa
  . . .
  LXI D, <dirección>     ; argumento en el par de registros DE
  MVI C, <número>        ; número de la función en el registro C
  CALL BDOS              ; llamada al servicio monitor.
```

Analicemos con más detalle el conjunto de servicios ofrecidos por el BDOS. Podemos clasificarlos en cuatro grupos: operaciones sobre los elementos asíncronos (consola, impresora, lectora rápida, etc...), operaciones sobre los discos, operaciones sobre los ficheros y operaciones diversas.

Algunas funciones, en particular las relativas a los conceptos de número de usuario, de atributo de fichero o de acceso directo, sólo están disponibles en CP/M 2.2.

1.6.1.2. Reinicialización del sistema

Existen dos métodos para la terminación normal de un programa que se ejecuta en el área TPA. La primera consiste en provocar una reinicialización del sistema con una bifurcación a la dirección 0000H o con una llamada al BDOS con el código de función C = 0. Ambas posibilidades son totalmente equivalentes.

```
BOOT   EQU    0000H    ou    BDOS    EQU    0005H
       ...                           ...
       ...                           MVI    C,0
       JMP    BOOT                   CALL   BDOS
```

Un segundo método, más rápido y factible con aquellos programas que no se solapen con el CCP, consiste en restaurar el valor del apuntador de pila al comienzo del programa y ejecutar una instrucción RET.

```
OLDSP  DS     2        ; salvar el SP (STACK POINTER)
DEB    LXI    H,0      ; valor del puntero de pila al comienzo
       DAD    SP       ;
       SHLD   OLDSP    ; salvado en la variable OLDSP
       ...
EXIT   LHLD   OLDSP    ; restauración del puntero de pila
       SPHL            ; en el registro SP
       RET             ; retorno al programa llamador CCP
```

1.6.1.3. Operaciones sobre la consola y la impresora

– Lectura de un carácter desde la consola (Console Input)

llamada: C = 1
retorno: A = carácter en ASCII

El carácter se envía a la consola (modalidad eco), excepto el <lf> si va a continuación de un <cr>. Los caracteres de tabulación se sustituyen por espacios por grupos de ocho columnas. Si se trata de un carácter de control, exceptuados <cr> y <lf>, se envía con eco a la consola precedido del carácter "↑"

– Escritura de un carácter en consola (Console output):

llamada: C = 2
retorno: E = carácter en ASCII

Los carácteres de tabulación se extienden con espacios por grupos de ocho columnas. Si se teclea el carácter <ctrl-S> durante una salida por consola, dicha salida queda detenida hasta que se teclea cualquier otro carácter.

– Lectura de un carácter desde la lectora (Reader input):

llamada: C = 3
retorno: A = carácter en ASCII

La lectura se efectúa sobre el periférico asignado a la lectora RDR: (ver mandato STAT y el octeto IOBYTE).

– Escritura de un carácter sobre la perforadora (Punch output):

llamada: C = 4
retorno: E = carácter en ASCII

Se transmite el carácter ASCII del registro E hacia el periférico asignado a PUN:.

– Escritura de un carácter sobre la impresora (List output):

llamada: C = 5
retorno: E = carácter en ASCII

Se envía el carácter ASCII del registro E hacia el periférico asignado a LST: (impresora).

– Entrada-salida directa sobre la consola (Direct console I/O):

llamada:	C = 6
	E = 0FFH lectura de un carácter
	E = 0FEH estado de la puerta asíncrona
	E = carácter ASCII a escribir
retorno:	A = carácter si lectura
	A = status si se solicita el estado

Las entradas-salidas directamente sobre la consola se utilizan para aquellas aplicaciones que requieran tiempos de respuesta muy cortos. No se filtran los caracteres de control y se deja su gestión al criterio del usuario.

– Lectura del octeto IOBYTE (Get I/O byte):

llamada: C = 7
retorno: A = octeto IOBYTE

El octeto IOBYTE, que define las asignaciones de los periféricos físicos a los periféricos lógicos, es devuelto en el acumulador A.

– Modificación del octeto IOBYTE (Set I/O byte):

llamada: C = 8
retorno: E = nuevo octeto IOBYTE

– Visualización de un mensaje en consola (Print String):

retorno: DE = dirección de la cadena de caracteres

Se visualizan los caracteres a partir de la dirección definida por el par de registros DE hasta que se detecta un carácter nulo 00H o un carácter "$".

Ejemplo:

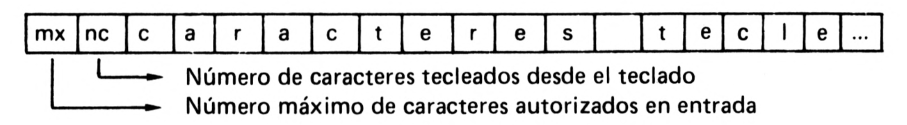
```
BDOS  EQU   5
MESS  DB    ' A SU SERVICIO, 0DH,0AH,'$'
      ...
      MVI   C,9
      LXI   D,MESS
      CALL  BDOS
```

– Lectura desde consola de una línea completa (Read console buffer):

llamada: C = 10 ó 0AH
DE = dirección del buffer de recepción del usuario; el número máximo de caracteres admitidos en lectura debe situarse en el primer octeto del buffer.

retorno: el número de caracteres tecleados se escribe en buffer+1; los caracteres tecleados se escriben a partir de buffer+2.

Formato del buffer de lectura:



Esta llamada permite rellenar directamente un buffer de memoria con una línea escrita desde el teclado y que termina con un <cr>. El eco de los caracteres introducidos se envía a consola. Se extienden los tabuladores y se tienen en cuenta los caracteres especiales: borrado de un carácter, de una línea, reimpresión de una línea...

– Investigación del "status" de la consola (Get console status):

llamada: C = 11 ó 0BH
retorno: A = 0 0 Z-flag=1 si no hay carácter
A = FF 0 Z-flag si hay un carácter

Esta función es muy útil para comprobar si hay un carácter en el port de la consola.

1.6.1.4. Operaciones sobre los discos

– Reinicialización del disco de sistema (Reset disk system):

Llamada: C = 13 ó 0DH

Esta función se utiliza para volver a dejar el disco de sistema en un estado normal: protección R/W y dirección del DMA en la posición 0080H de la memoria después de un cambio de disco sin operación de arranque en frío.

– Selección de un disco (Select disk):

llamada: C = 14 ó 0EH
E = número de disco a seleccionar

Los números de los discos 0, 1, ..., 15 corresponden a los nombres de los discos A, B, ..., P.

– Investigación de los discos disponibles (Return login vector):

llamada: C = 24 ó 18H
retorno: HL = vector de login

El vector de "login" es una palabra de 16 bits en la que cada bit corresponde a un disco (0 a 15). El bit de menor peso del registro L está asociado al disco A y así sucesivamente hasta el bit de mayor peso que está asociado al disco P. El bit indica que el disco no está conectado (bit = 0) o que sí lo está (bit = 1).

– Número del disco actual (Return current disk):

llamada: C = 25 ó 19H
retorno: A = número del disco actual

Esta función devuelve un número de 0 a 15 correspondiente al disco actualmente seleccionado.

– Protección de un disco (Write protect disk):

llamada: C = 28 ó 1CH

Esta función protege contra escritura la totalidad del disco actual. Su desprotección se hace efectiva en la siguiente reinicialización del sistema (arranque en caliente o en frío).

– Lectura del vector de protección (Get Read/Only vector):

llamada: C = 29 ó 1DH
retorno: HL = vector R/O

Esta función devuelve un vector de 16 bits en el que cada bit está asociado a un disco (A peso débil y P peso fuerte). Un bit a 1 indica la protección temporal del disco asociado.

1.6.1.5. Operaciones sobre los ficheros

– Apertura de un fichero ya existente (Open file):

llamada: C = 15 ó 0FH
DE = dirección del FCB
retorno: A = código de directorio (0,1,2,3) si fichero encontrado
A = 0FFH si fichero no encontrado.

Se realiza un barrido del directorio comparando las posiciones 1 a 14 del FCB con las de cada entrada. Si el FCB contiene caracteres "?" en el campo simbólico del nombre del fichero, el barrido se detiene cuando se detecta la primera coincidencia. El código devuelto en el acumulador A corresponde al índice de la entrada dentro del sector actual del directorio (0, 1, 2, 3). Antes de hacer un "open", el registro actual "cr" del FCB debe ser puesto a ceros por el programador para permitir el acceso secuencial al fichero desde el primer registro.

– Cierre de un fichero (Close file):

llamada: C = 16 ó 10H
DE = dirección del FCB
retorno: A = código de directorio (0, 1, 2, 3) si fichero existe
A = 0FFH si fichero no existe.

El cierre de un fichero en lectura no es indispensable; por el contrario, se debe cerrar un fichero en creación o en actualización para que los registros escritos en disco puedan ser contabilizados en el "map" de entrada del fichero. Los códigos devueltos son idénticos a los de la función "open".

– Búsqueda del primer fichero ambiguo (Search for first):

llamada: C = 17 ó 11H
DE = dirección del FCB
retorno: A = código de directorio (0,1,2,3) si existe coincidencia
A = 0FFH si no se encuentra ningún fichero

Los caracteres "?" del nombre del fichero del FCB expresan que el fichero es ambiguo. Si se detecta una coincidencia durante el barrido del directorio, se devuelve el índice de la entrada correspondiente en el registro A. El sector actual del directorio se encuentra en la dirección 0080H de la memoria.

– Búsqueda del siguiente fichero ambiguo (Search for next):

llamada: C = 18 ó 12H
DE = dirección del FCB
retorno: idéntico a la función de búsqueda del primer fichero

– Destrucción de un fichero (Delete file):

llamada: C = 19 ó 13H
DE = dirección del FCB
retorno: A = código de directorio (0, 1, 2, 3) si el fichero existía
A = 0FFH si el fichero no existe

Esta llamada destruye todas las entradas, si el fichero tiene varias extensiones, del directorio del disco. Todos los bloques utilizados por el fichero quedan disponibles.

– Lectura secuencial de un registro (Read sequential):

llamada: C = 20 ó 14H
DE = dirección del FCB
retorno: A = 0 si la lectura se ha realizado correctamente
A = 1 si se produce un error de lectura

Los 128 octetos del registro actual del fichero son leídos en el buffer de disco DMA. El contador de registros "cr" se incrementa en una unidad con cada transferencia; en caso de desbordamiento, se abre la extensión "ex" siguiente del fichero y se continúa la lectura con un contador de registros puesto a ceros.

– Escritura secuencial de un registro (Write sequential):

llamada: C = 21 ó 15H
DE = dirección del FCB
retorno: A = 0 si la escritura se ha realizado correctamente
A = distinto de cero si error de escritura o saturación de disco.

Esta función escribe un registro de 128 octetos, memorizado a partir de la dirección DMA, a continuación del registro anterior. El contador de registros "cr" se incrementa con cada escritura; si existe desbordamiento, se abre automáticamente una extensión "ex" en el directorio y el contador "cr" parte de cero.

– Creación de un fichero (Make file):

llamada: C = 22 ó 16H
DE = dirección del FCB
retorno: A = código de directorio (0,1,2,3) si normal
A = 0FFH si el directorio está saturado.

Esta función es similar a la función "open" con la única diferencia de que el fichero se crea aunque el fichero ya exista. Se inicializa una entrada en el directorio de manera que el fichero esté vacío.

– Cambio de nombre de un fichero (Rename file):

llamada: C = 23 ó 17H
DE = dirección del FCB
retorno: A = código de directorio (0,1,2,3) si el fichero existe
A = 0FFH si el fichero no existe

Esta llamada cambia el nombre y el tipo del antiguo fichero, especificados en los 16 primeros octetos del FCB, por el nombre y el tipo del nuevo fichero, especificados en los 16 octetos siguientes.

– Inicialización de la dirección del buffer de disco (Set DMA address):

llamada: C = 26 ó 1AH
DE = dirección del buffer de disco DMA

Aunque la configuración física no soporte el mecanismo de acceso directo a memoria (Direct Memory Access), el buffer DMA es un área de memoria de 128 octetos destinada a albergar los datos que se reciben del disco, o los que se van a grabar en disco. El programador dispone de la dirección DMA implícita en la dirección 0080H de la memoria que es forzada en cada reinicialización del sistema.

– Modificación de los atributos de un fichero (Set file attributes):

llamada: C = 30 ó 1EH
DE = dirección del FCB
retorno: A = código de directorio

Los atributos de fichero (R/O, R/W) y (DIR, SYS) están representados por el bit de mayor peso de los octetos t1 y t2 del tipo de fichero del FCB. El tipo de fichero no queda modificado porque los caracteres ASCII están codificados sobre los siete bits de menor peso.

– Lectura directa de un registro (Read random):

llamada: C = 33 ó 21H
DE = dirección del FCB
retorno: A = código de error

La operación de lectura del registro especificado se realiza sobre los octetos "r0", "r1" y "r2" de las posiciones 33, 34 y 35 del FCB. La dirección del registro está pues codificada en 24 bits para poder alcanzar la capacidad máxima de 8 Mega-octetos. El octeto "r0" representa los pesos débiles y el octeto "r2" los pesos fuertes.

– Escritura directa de un registro (Write random):

llamada: C = 34 ó 22H
DE = dirección del FCB
retorno: A = código de error

La escritura directa funciona según los mismos principios ya enunciados para la lectura directa.

– Cálculo del tamaño de un fichero directo (Compute file size):

llamada:	C = 35 ó 23H
	DE = dirección del FCB
retorno:	el campo r0, r1, r2 del FCB queda inicializado

El tamaño virtual de un fichero construido en forma secuencial es igual a su tamaño físico. Si el fichero está construido en modo directo aparecerán "huecos" en el espacio asignado y el tamaño útil no será el mismo que el tamaño solicitado.

– Posicionamiento sobre un registro directo (Set random record):

llamada:	C = 36 ó 24H
	DE = dirección del FCB
retorno:	el campo r0, r1, r2, del FCB queda inicializado

Esta función permite posicionarse sobre un determinado registro de un fichero construido en modo secuencial.

1.6.1.6. Operaciones diversas

– Número de versión CP/M (Return versión number):

llamada:	C = 12 ó 0CH
retorno:	HL = número de versión

El registro H indica si se trata de un MP/M (H = 01) o de un CP/M (H = 00). Si el registro L es nulo indica que se trata de versiones anteriores al CP/M 2,0, en caso contrario contiene los valores 20 H, 21H,..., 2FH para las diferentes versiones del CP/M 2.0.

– Obtención de las direcciones de asignación (Get addr alloc):

llamada:	C = 27 ó 1BH
retorno:	HL = dirección del vector de asignación

En memoria principal se mantiene un vector de asignación para cada disco conectado. Este vector permite conocer, por ejemplo, el tamaño que queda disponible de almacenamiento en disco (mandato STAT).

– Obtención de la dirección de los parámetros de los discos (Get addr Disk parms):

llamada: C = 31 ó 1 FH
retorno: HL = dirección del DPB (bloque de parámetros del disco)

Esta función devuelve la dirección de una tabla en la que aparecen las características de los discos de la configuración: simple o doble densidad, número de caras, número de pistas, número de sectores por pista, tamaño del sector,...

– Inicialización o lectura del número de usuario (Set/Get user code):

llamada: C = 32 ó 20H
E = 0FFH obtención del número de usuario
E = código de usuario para modificación
retorno: A = número de usuario si obtención

Esta función permite la lectura del número del usuario actual o su modificación. Se permite un máximo de dieciséis usuarios bajo CP/M 2.2, a los cuales se les asigna los números del 0 al 15.

1.6.2. Las primitivas de sistema del BIOS

1.6.2.1. Los puntos de entrada de las primitivas de sistema

Al contrario de lo que ocurre con el BDOS en el que el acceso a las funciones se hace a través de un paso obligado por la dirección 0005H de memoria, el acceso a las primitivas de entradas-salidas del BIOS se hace mediante una tabla o vector de bifurcación situado en la base del BIOS por razones de comodidad de interfase con los otros módulos (CCP y BDOS). La base del BIOS se deduce a partir del contenido de la dirección 0001H de memoria que apunta hacia el punto de entrada del "warm star" en BIOS+3.

Este vector de bifurcación consta de 17 instrucciones de bifurcación hacia las rutinas específicas del BIOS. Podemos clasificar a dichos programas en tres categorías:

– reinicialización del sistema,
– entradas-salidas modalidad carácter,
– entradas-salidas disco.

Las principales funciones tratadas por el BIOS son las siguientes:

BIOS	JMP boot	: reinicialización después de un "cold start"
BIOS+3	JMP wboot	: reinicialización después de un "warm start"
BIOS+6	JMP const	: verificación de "status" de consola
BIOS+9	JMP conin	: lectura de un carácter desde la consola
BIOS+12	JMP conout	: escritura de un carácter en la consola
BIOS+15	JMP list	: escritura de un carácter sobre la impresora
BIOS+18	JMP punch	: escritura de un carácter sobre la "perforadora"
BIOS+21	JMP reader	: lectura de un carácter desde la lectora
BIOS+24	JMP home	: posicionamiento sobre la pista 00 del disco
BIOS+27	JMP seldsk	: selección de un disco "drive"
BIOS+30	JMP settrk	: inicializa el número de pista
BIOS+33	JMP setsec	: inicializa el número de sector
BIOS+36	JMP setdma	: inicializa la dirección del buffer DMA
BIOS+39	JMP read	: lectura de un sector del disco
BIOS+42	JMP write	: escritura de un sector del disco
BIOS+45	JMP listst	: verificación del "status" de la impresora
BIOS+48	JMP sectran	: conversión de números de sectores

1.6.2.2. Primitivas de reinicialización

BOOT Es llamado solamente por el cargador del sistema operativo durante el arranque en frío, es decir, durante la operación de "bootstrap" después de una "RAZ" manual del sistema. Se inicializan algunas variables del área SPA en la zona baja de la memoria y el control se cede al módulo de diálogo CCP.

WBOOT Es llamado en cada reinicialización del sistema, ya sea provocada por el tecleo de un carácter <ctrl-C> ya sea programada (JMP 000H o CALL 0005H con C ∠ 0). Su misión consiste en volver a traer a memoria el CP/M y ceder el control al CCP.

1.6.2.3. Primitivas de entradas-salidas modalidad carácter

CONST devuelve en el acumulador el "status" del periférico asignado a la consola. Si A = 0 no está presente ningún carácter, si A = 0FFH está presente un carácter.

CONIN — Lee un carácter desde el periférico consola en el registro A. Se suprime el bit de paridad.

CONOUT — Envía el carácter escrito en el registro "C" hacia la consola.

LIST — Envía el carácter escrito en el registro "C" hacia el periférico asignado a la impresora.

PUNCH — Envía el carácter escrito en el registro "C" hacia el periférico asignado a la perforadora rápida.

READER — Lee un carácter desde el periférico asignado al lector rápido en el registro A.

1.6.2.4. Primitivas de entradas-salidas disco

HOME — Coloca las cabezas de lectura del disco seleccionado, al comienzo del disco, sobre la pista 0.

SELDSK — Selecciona el disco indicado por el registro C: 0 para A, 1 para B, 15 para P.

SETTRK — Permite posicionarse sobre el número de pista indicado por el par de registros BC.

SETSEC — Permite seleccionar el número de sector dentro de la pista, sobre el que se realizará la transferencia; este número está en el registro C.

SETDMA — Permite inicializar la dirección DMA contenida en el par de registros BC.

READ — Se posiciona sobre los números de pista y sector definidos con anterioridad con SETTRK y SETSEC, y lee el sector para transferir su contenido a la dirección inicializada por SETDMA. READ devuelve un código de terminación en el acumulador A; si A = 0 la lectura se ha realizado correctamente, en caso contrario, si A = 1 se ha producido un error irrecuperable.

WRITE — Escribe los datos, 128 octetos de memoria, situados en la dirección DMA, en el sector especificado.

SECTRAN Calcula el número de sector físico a partir del número del sector lógico dado por el registro C. En efecto, por razones de eficacia, en algunos discos, los sectores lógicos no están contiguos para evitar tener que dar una vuelta completa al disco cuando deseamos posicionarnos en el sector siguiente. Se aplica un factor igual a 6 en la numeración de los sectores. Los primeros sectores lógicos (1,2,3,4...) corresponden por ejemplo a los sectores físicos (1,7,13, 19...). Generalmente existe una tabla de traducción del BIOS por tipo de disco; su dirección viene dada por el par DE y el resultado es devuelto en el par HL.

2

El sistema MP/M

2.1. GENERALIDADES SOBRE EL MP/M

El sistema operativo MP/M (Multi-Programming Monitor) es un sistema "multi-usuario", basado en el CP/M que es un sistema mono-usuario.

El propósito de este sistema es soportar un acceso multi-terminal con posibilidad de "multi-tasking" (multi-tarea) a nivel de cada terminal.

MP/M puede gestionar de 1 a 16 usuarios, estando cada uno de ellos asociado a un puesto de trabajo: una consola física.

Bajo MP/M aparece igualmente el concepto de usuario, materializado por un número de usuario de 0 a 15 (no confundir el concepto de puesto de trabajo o consola con el de propietario o número de usuario).

Todo usuario, del 1 al 15, tiene acceso en lectura y en ejecución a todos los programas de utilidades y a las bibliotecas de subprogramas almacenadas bajo el número de usuario común 0 (número del sistema). Por el contrario, de un usuario a otro, los programas son independientes y dos usuarios pueden crear ficheros con el mismo nombre sin que por ello exista peligro de que se entremezclen. Cada usuario es propietario de sus datos y no interfiere con los de su vecino.

Los programas de utilidades, normalmente utilizados bajo el número de usuario común, permiten a los usuarios compartir mandatos evitando así la ocupación de su propio espacio del disco por las duplicaciones de los programas.

El paso de un usuario a otro se hace por un mecanismo de compartición del tiempo de la CPU llamado "Time-slicing". A cada programa activo (processus) se le asigna una porción de tiempo de 20 mili-segundos. Para mejorar el reparto del tiempo entre los procesos, todo proceso que inicialice una entrada-salida o solicite una función monitor, queda desactivado en beneficio del proceso en espera, que tenga la prioridad más alta.

Todo sucede como si cada usuario trabajase él sólo con la máquina. Un inconveniente normal pero inevitable subsiste: a mayor número de usuarios o de procesos, mayor degradación del tiempo de respuesta (overhead-time).

Todo sucede como si cada usuario trabajase él sólo con la máquina. Un inconveniente normal pero inevitable subsiste: a mayor número d de usuarios o de procesos, mayor degradación del tiempo de respuesta (overhead-time).

2.2. ENTORNO FISICO

Las configuraciones físicas con las que puede funcionar el MP/M son las siguientes:

- microprocesadores: 8080, 8085 ó Z80
- memoria: de 32 a 400 K-octetos, con o sin bancos
- consolas: 1 a 16 consolas tipo "CRT" (pantalla + teclado)
- discos: 1 a 16 discos flexibles o duros
- base de tiempos: reloj programable bajo interrupciones

2.3. DESCRIPCION FUNCIONAL DEL MP/M

El sistema operativo está construido alrededor de un núcleo de tiempo real multi-tarea. Este núcleo lógico es independiente del entorno físico, pero depende de la CPU porque está escrito en lenguaje ensamblador 8080.

El sistema MP/M se divide en tres partes representando cada una de ellas un nivel monitor.

— **XIOS** **(eXtended Input Output Supervisor)**
— **BDOS** **(Basic Disk Operating System)**
— **XDOS** **(eXtended Disk Operating System)**

A este conjunto de módulos hay que añadir una zona reservada a la interfase (comunicación) sistema-usuario. El tamaño de esta zona es variable dependiendo de la configuración adoptada. Se encuentra en las direcciones altas de la memoria y se subdivide en tres partes:

SYSTEM.DAT	:	parámetros de generación
CONSOLE.DAT	:	tampones de entrada-salida de las consolas
USERSYS.STK	:	pila de trabajo

Asímismo, el MP/M permite, si el monitor cabe en 16 K, asignar el resto de la memoria común a los procesos de los usuarios declarados como residentes durante la generación del sistema. Se trata de los programas tipo "RSP" (procesos del sistema residentes).

El resto de la memoria disponible es lo que ya, en el CP/M, hemos denominado la "TPA".

2.3.1. El XIOS

El módulo XIOS no es más que una extensión del módulo BIOS bajo CP/M. Contiene todos los programas específicos (drivers) ligados al entorno físico (hardware). El XIOS constituye la interfase software-hardware entre los otros módulos del sistema y los controladores de los periféricos.

Las principales primitivas programadas permiten:

– la gestión de las puertas serie y paralelo;
– la selección de disco, cara, pista y sector;
– la selección y protección de los bancos de memoria;
– la inicialización de las transferencias físicas sobre disco;
– el tratamiento de las interrupciones;
– la gestión de la base de tiempos (contador)

Estos subprogramas pueden ser modificados, cambiados o suprimidos a gusto del usuario, según la configuración que se desee adoptar (se suministra el "fuente").

2.3.2. El BDOS

El módulo "BDOS" contiene las funciones lógicas de gestión de los ficheros de disco y de las consolas conversacionales (gestión del procedimiento asíncrono).

Este módulo trata todas las "llamadas del sistema" disponibles para los usuarios. De hecho, existe un punto de entrada en la base de memoria (dirección 0005H) para permitir el acceso a cualquier servicio monitor.

Bajo CP/M, el módulo "BDOS" está completamente residente; por el contrario, bajo MP/M, en el caso de algunas configuraciones de memorias organizadas en bancos, el módulo "BDOS" se divide en dos partes:

– ODOS : parte re-entrante en la memoria común gestión del diálogo con la consola;

– BNKBDOS : parte no-reentrante que se extiende sobre el banco cero gestión de los ficheros.

2.3.3. El XDOS

El módulo XDOS contiene el núcleo lógico del MP/M, así como las extensiones para la gestión de los ficheros.

El módulo XDOS comprende las siguientes funciones:

– gestión de los procesos por el distribuidor "dispatcher";
– gestión de las colas o ficheros de esperas;
– gestión de los sucesos o flags;
– gestión de la memoria;
– gestión de la base de tiempos;
– el proceso "terminal" (TMP)
– el intérprete de mandatos (CLI), con el cargador
– las funciones suplementarias del MP/M.

2.3.3.1. Gestión de los procesos

El recurso "CPU" es asignado al candidato que tenga la prioridad más alta. La determinación de la prioridad viene dada por el módulo de gestión de los procesos llamado "dispatcher" o distribuidor. Cada

proceso posee un descriptor de proceso "PD"; consiste en una tabla que contiene todas las informaciones que el sistema precisa para conocer las características propias del proceso:

- número del usuario
- número de la consola
- número del disco actual
- prioridad del proceso
- nombre del proceso
- segmento de memoria utilizado
- zona de trabajo: cadenas, bufferes, pila, etc...

El "dispatcher" utiliza estas informaciones para salvar el estado del proceso en curso, determinar cuál es el proceso que hay que ejecutar a continuación y restaurar el estado del proceso cuando se le vuelva a ceder el control.

El módulo de gestión de los procesos es llamado:

- con cada llamada al sistema (CALL BDOS)
- con cada interrupción externa
- con cada ciclo ("top") del reloj (cada 20 ms)

A los procesos que tienen la misma prioridad, el distribuidor los trata cíclicamente (round-robin) otorgándoles porciones iguales de tiempo de la CPU (slices).

2.3.3.2 Gestión de las colas

Las colas o ficheros de espera (FIFO: First In First Out) se utilizan principalmente para el intercambio de mensajes entre los procesos, para sincronizar los procesos y para resolver la exclusión mutua (acceso a un recurso único no compartible).

La gestión de las colas bajo MP/M está concebida tanto para los procesos del sistema como para los programas de los usuarios.

De hecho, las colas se tratan de forma similar a los ficheros en disco aunque estén siempre residentes en memoria. Las colas pueden ser creadas, abiertas, escritas, leídas y distribuidas.

Se puede leer o escribir una cola condicional o incondicionalmente. Si se efectúa una lectura sobre una cola en la que no está presente el mensaje, el "dispatcher" suspende el proceso llamador hasta que el mensaje sea escrito en la cola por cualquier otro proceso. Del mismo modo, si se escribe un mensaje, en forma incondicional, en una cola ya

llena, el proceso emisor esperará a que otro proceso lea los mensajes almacenados para que se pueda liberar el espacio necesitado por el mensaje en espera. Esto es lo que se llama sincronización de los procesos mediante el mecanismo del "productor-consumidor".

El sistema de colas garantiza también la exclusión mutua entre procesos concurrentes, asegurándose de que las fases críticas no van a ser interrumpidas durante su ejecución (colas de tipo MX).

La estructura de datos de una cola consta del bloque de control de cola "QCB" que debe estar residente y el bloque de usuario de control de cola "UQCB" situado en el programa de usuario.

Hay dos tipos de "QCB": las colas circulares y las colas encadenadas. Estos tipos dependen del tamaño del mensaje: los mensajes de 0 a 2 octetos utilizan las colas circulares mientras que los mensajes de 3 ó más octetos utilizan las colas encadenadas.

2.3.3.3. Gestión de los sucesos

El MP/M utiliza la gestión de los sucesos lógicos para sincronizar las tareas que han dado lugar a procesos asíncronos (entradas-salidas, por ejemplo). Los sucesos lógicos sustituyen a los sucesos físicos (interrupciones), porque el MP/M es independiente del entorno físico, y por lo tanto de las interrupciones, simulando de esta forma el entorno físico. El MP/M soporta 32 "flags" de tipo suceso.

Las operaciones que se pueden realizar con los sucesos son:

– espera de un suceso (Wait)
– activación de un suceso (Flag)

2.3.3.4. Gestión de la memoria

El MP/M gestiona la memoria en unos segmentos predefinidos en la generación del sistema. Se pueden controlar ocho segmentos de memoria de 0 a 64 Kilo-octetos, ya sea la memoria monolítica o estructurada en "bancos", con o sin protección. Los segmentos de memoria están descritos internamente por los descriptores de memoria MDF (Memory Descriptor).

Los bancos son unas particiones fijas y totalmente independientes. Observe que en el instante "t" la CPU sólo direcciona un banco, es decir, 16 + 48 Kilo-octetos. Los programas no son re-entrantes de una partición a otra.

2.3.3.5. Gestión del tiempo

La gestión del tiempo bajo MP/M se reparte según dos procesos: el proceso "Tick" y el proceso "Clock".

El proceso "Tick", activado cada 20 mili-segundos, determina el período asignado a cada uno de los procesos activos para la utilización de la CPU. Esta frecuencia no debe ser demasiado elevada, porque en ese caso se podría dar lugar a un aumento significativo del "overhead-time" debido a una conmutación excesiva de los procesos. A la inversa, con una frecuencia demasiado baja, los procesos retendrían el recurso CPU durante un período demasiado largo, penalizando de esta forma a los otros procesos.

El proceso "Clock" es activado por el proceso "Tick" cada segundo. Su misión consiste en mantener y proporcionar la fecha (día, mes, año, hora, minuto, segundo).

Además, las primitivas de sistema ofrecen la posibilidad de regular los programas que se desean cargar y ejecutar en un momento dado, elegido de antemano, así como la de desencadenar la ejecución de un proceso durante un período determinado.

2.3.4. Areas de trabajo del sistema

2.3.4.1. SYSTEM.DAT

Este segmento situado en la parte baja de la memoria ocupa 256 octetos (una página). Contiene las informaciones que necesita el "cargador del sistema" para reconfigurar el sistema dinámicamente. La introducción de estas informaciones la realiza el generador del sistema GENSYS.

00–00	última página de la memoria física (00 = toda la memoria)
01–01	número de consolas que se pueden conectar
02–02	número de interrupción reservada al debug DDT (breakpoint)
03–03	booleano de asignación de pila para los "CALL" de sistema
04–04	booleano indicando si la memoria está organizada en bancos
05–05	booleano indicando si la CPU es un Z80
06–06	booleano indicando si el BDOS está configurado en bancos (BNKBDOS)
10–2F	tabla de segmentos de memoria iniciales
30–4F	tabla de puntos de parada del DDT
50–6F	pila para las llamadas de sistema de los usuarios

Después de la carga, en este segmento se guardan determinados parámetros de funcionamiento.

2.3.4.2. CONSOLE.DAT

El tamaño de este segmento es proporcional al número de consolas definidas durante la generación del sistema. Cada consola requiere 256 octetos que contienen:

– èl descriptor de proceso terminal (TMP)
– una pila
– unos bufferes (tampones) para las entradas-salidas de la consola.

2.3.4.3. UNSERSYS.STK

Este segmento es opcional y contiene 64 octetos de espacio de pila por segmento de memoria de usuario. Se utiliza como una pila temporal cuando los programas de los usuarios efectúan llamadas al sistema (CALL BDOS).

2.4. ESTRUCTURA DEL SISTEMA MP/M

2.4.1. Estructura de la memoria

El MP/M es un sistema multi-usuario lo cual implica la presencia de varios programas en memoria.

El sistema ocupa un mínimo de 15 Kilo-octetos y un máximo de 20 Kilo-octetos de memoria. El resto de la memoria, dividida en segmentos (ocho como máximo), está reservada para los programas de los usuarios (segmento = TPA). Cada segmento de memoria está, a su vez, dividido en dos regiones: la SPA (System Parameter Area) y la UCA (User Code Area).

2.4.1.1. La SPA

La primera región, la SPA (área de parámetros del sistema), ocupa los 256 octetos del segmento de memoria. A este área también se le llama “página de base del segmento”.

2.4.1.2. La UCA

La segunda región del segmento de memoria del usuario, la UCA (área de código del usuario), comienza en la dirección 0100H relativa a la base del segmento de memoria. Cuando se carga un programa, su código se escribe en el segmento de memoria, a partir de la UCA.

Los programas de usuario se cargan en memoria mediante el intérprete de mandatos "CLI (Command Line Interpreter). CLI recibe los mandatos del "TMP" (Terminal Message Process) leídos desde la entrada de consola.

El TMP es un programa re-entrante con el que los usuarios se pueden comunicar tecleando determinadas líneas de mandatos.

Cada mandato está precedido por un prefijo "prompt" o vector de "login" que indica el número de usuario, seguido del nombre del disco actual. Estas son las diferentes formas sintácticas aceptadas para un mandato:

```
mandato
mandato fichero1
mandato fichero1 fichero2
```

donde "mandato" puede ser o una "cola" o un programa de usuario.

Cuando el CLI recibe un mandato, realiza un análisis sintáctico de la primera parte e intenta abrir la "cola". Si la apertura de la "cola" es satisfactoria, se copia en la "cola" la continuación del mandato y termina la operación del CLI.

Si la "cola" no existe, CLI intenta abrir un fichero tipo "PRL" (programa reubicable) cuyo nombre será el del mandato. Si la apertura del fichero es satisfactoria, se lee la cabecera del fichero PRL para determinar los argumentos de memoria (tamaño del programa, por ejemplo). Se realiza una solicitud de asignación de memoria reubicable para obtener un segmento de memoria en el que se cargará y ejecutará el programa. Si se satisface esta solicitud, el fichero PRL se carga en el segmento y se ejecuta, terminando así la operación del CLI.

Si el fichero PRL no existe, el CLI intenta abrir un fichero del mismo nombre pero de tipo COM. Si la apertura es satisfactoria, se solicita una asignación de memoria absoluta, a partir de la TPA con base en la dirección 0100H. Si se satisface esta solicitud, se lee el fichero tipo COM en el segmento absoluto de la TPA y se ejecuta, terminando el proceso CLI.

Para cada programa cargado, CLI crea un descriptor de proceso, inicializa una pila de trabajo y le otorga una prioridad implícita.

2.4.2. Estructura del disco

Al igual que en el CP/M, las dos primeras pistas del disco (pistas 0 y 1) están reservadas, pero el sistema MP/M, demasiado voluminoso, no cabe en ellas por lo que está situado en un fichero llamado "MPM. SYS". Este fichero representa la imagen exacta de memoria del MP/M que se cargará durante la operación de arranque en frío (cold start o cold boot).

Las dos pistas reservadas contienen un "bootstrap" en el sector 1 y el cargador del sistema MP/M, que está dividido en dos partes: MPMLDR y LDRBIOS.

MPMLDR es el cargador lógico del sistema MP/M, independiente del entorno físico, que carga el fichero "MPM.SYS" en memoria y adapta su configuración con la ayuda del fichero "SYSTEM.DAT".

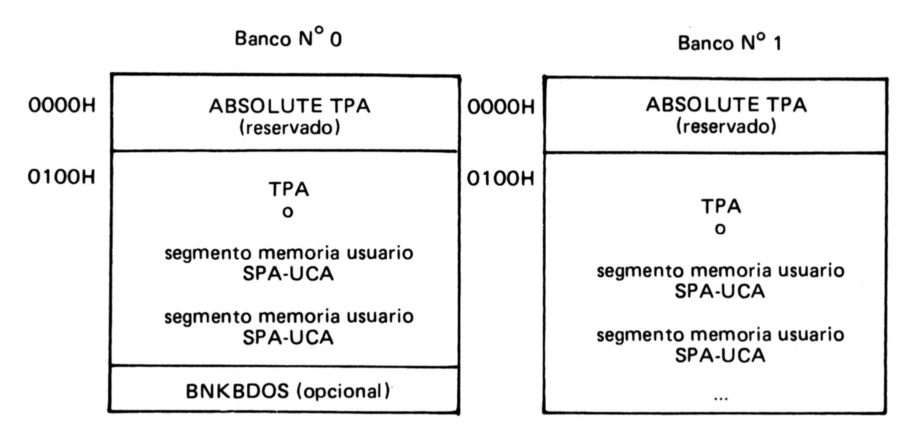
LDRBIOS es, por el contrario, un "mini-BIOS", dependiente del entorno físico (disco flexible o duro, sectorizado "soft" o "hard"). Las primitivas de LDRBIOS, situadas en direcciones concretas son utilizadas por MPMLDR.

Pista	Sector	Dirección memoria	Nombre del módulo
00	01	Dirección del boot	BOOT
00 .. 00	02 .. 25	0100H 0C80H	MPMLDR
00 01 .. 01	26 01 .. 19	0D00H 0D80H 1680H	LDRBDOS
01 .. 01	20 .. 26	1700H 1A00H	LDRBIOS

Fig. 10.— Implantación en disco de los módulos de carga.

2.5. LOS MANDATOS BAJO CP/M

Se ha mantenido la compatibilidad entre CP/M y MP/M a nivel de diálogo hombre-máquina. Como resultado, todos los mandatos que estaban disponibles bajo CP/M siguen siendo válidos bajo MP/M.



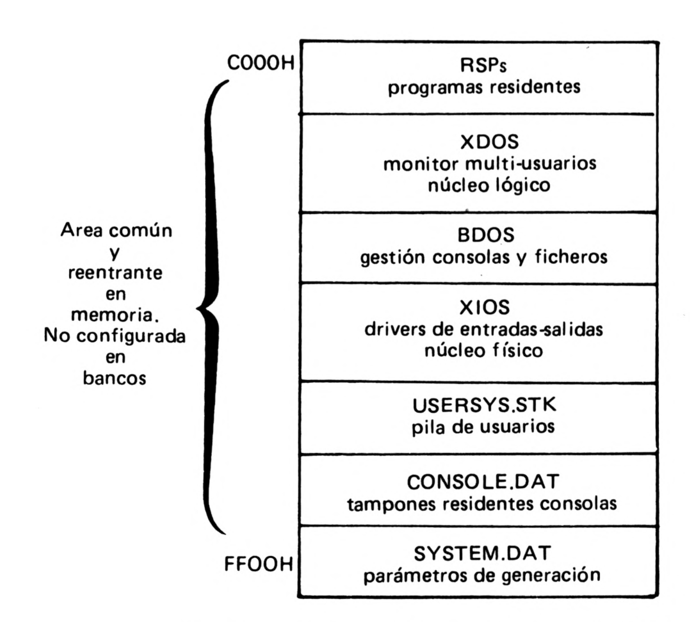


Fig. 11.— Implantación en memoria del MP/M.

No obstante existe una diferencia ya que el MP/M puede aceptar dos tipos de mandatos: los mandatos de tipo COM, enteramente compatibles con los del CP/M, que se ejecutan a partir de la dirección 0100H de la memoria, en la TPA, y los mandatos de tipo PRL (Program relocatable), mejor adaptados al contexto MP/M y que se ejecutan en uno de los segmentos definidos durante la generación del sistema. Los programas de utilidades permiten convertir ficheros ejecutables tipo HEX en tipo PRL (GENMOD) o de tipo PRL en tipo COM (PRLCOM).

El arranque de la ejecución de un programa bajo MP/M se hace tecleando su nombre después de haber recibido el prompt de invitación, llamado también "vector de login", que consta del número de usuario seguido por el nombre del disco. El "prompt" termina siempre con el carácter ">".

Ejemplo:

0A>mandato

El carácter de función <ctrl-C>, escrito desde el teclado, permite abandonar automáticamente la ejecución del programa en curso (también se dice "abortar" un programa).

El carácter de función <crtl-D> da al usuario la posibilidad de desconectar de la consola el programa en curso. El usuario puede solicitar entonces la ejecución de otro programa con la condición de que el primero, que se ejecuta normalmente, controle el estado (status) de la consola. Para retornar al programa anteriormente desconectado basta teclear el mandato ATTACH seguido del nombre del programa a enganchar a la consola.

Ejemplo:

0A>PIP	llamada a PIP
*	prefijo de PIP
↑D	desconectar PIP
0A>PROG	llamada a PROG
. . . .	ejecución de PROG
0A>ATTACH PIP	PIP se conecta de nuevo con consola
*	prefijo de PIP

2.5.1. Los mandatos compatibles CP/M

Como ya hemos visto anteriormente, la mayoría de los mandatos CP/M, integrados al CCP o no residentes (ficheros COM), han sido transportados al MP/M, con el tipo PRL.

DIR
ERA
REN
TYPE
STAT
PIP
ED
LOAD
DDT
SUBMIT
DUMP

2.5.2. Los mandatos suplementarios del MP/M

En relación al sistema CP/M, el MP/M aporta unos programas de utilidades nuevos que están ligados a la característica propia de un contexto multi-usuario: CONSOLE, USER, DSKRESET, ERAQ, DDT–MPM, MPMSTAT, TOD, ABORT, SPOOL, STOPSPLR, SCHED, PRLCOM, GENHEX, GENMOD, GENSYS, MPMLDR.

2.5.2.1. CONSOLE

CONSOLE permite visualizar el número de la consola actual.

Ejemplo:

```
0A>CONSOLE
Console = 1
0A>
```

2.5.2.2. USER

Este mandato se utiliza para visualizar el número de usuario actual o para cambiar de un usuario a otro. En los dos casos aparece en pantalla el "prompt" o prefijo de invitación para teclear un mandato.

Ejemplo:

```
0A>USER
User Number = 0
0A>USER 5
User Number = 5
5A>                                     Prefijo del usuario núm. 5
```

2.5.2.3. DSKRESET

Este mandato se utiliza después de un cambio de disco, en particular de disquete. De hecho, cuando se introduce un nuevo disquete queda automáticamente protegido contra escritura (R/O), DSKRESET efectúa un "Reset Disk" que devuelve el status R/W al disquete siempre y cuando no lo utilice ningún otro usuario. En caso contrario, esta operación no tiene efecto alguno y se visualiza el siguiente mensaje:

```
Disk reset denied, Drive "d:" Console "c" Program "pppp"
```

donde "d" es el nombre del disco, "c" es el número de consola y "pppp" el nombre del programa que se está ejecutando.

Si no se da ningún argumento, se reinicializa el conjunto de discos conectados, en caso contrario los argumentos representan los nombres de los discos separados por una coma.

Ejemplo:

```
0A>DSKRESET
0A>DSKRESET A:, B:
```

2.5.2.4. ERAQ

Destrucción de uno o varios ficheros con validación por parte del usuario.

Ejemplo:

```
0A>ERAQ B:*.BAK
   B:ESSAI   BAK ?Y
   B:TEST    BAK ?Y
   B:SOURCE  BAK ?N
   0A>
```

2.5.2.5. DDT

El "debugger" DDT del MP/M conserva toda la funcionalidad ofrecida por el DDT del CP/M con los cuatro mandatos adicionales siguientes: W, V, N y B.

W (Write) Wn n = número de sectores a escribir

Este mandato sustituye al mandato SAVE que no existe bajo MP/M. El valor "n" puede calcularse con el mandato V. Para que se realice la escritura, es aconsejable utilizar el mandato "Inombre del fichero" que especifica sobre qué fichero hay que efectuar el salvado de la imagen de memoria.

Ejemplo:

```
-ISAUVE.COM
-W8
```

V (Value) V

Vv v = tamaño del programa en octetos

En el primer formato (sólo V) se devuelve la dirección del NEXT y puede servir para el segundo formato. Este devuelve el número de sectores correspondientes al programa, que puede ser utilizado a continuación por el mandato W.

Ejemplo:

```
-V
NEXT PC
0280 0100
-V280
0003
```

N (Normalize) N

Este mandato normaliza un fichero ejecutable de tipo PRL que ha sido leído por el mandato R.

B (Bitmap) Ba,0 ó Ba,1 a = dirección dentro del "MAP"

Este mandato permite modificar, poner a cero o a uno, los bits del "MAP" de un fichero reubicable de tipo PRL.

2.5.2.6. MPMSTAT

Visualización del estado actual del sistema en funcionamiento, en particular: número de consolas activas, número de procesos, estado de los procesos, estado de las colas, estado de los sucesos, distribución de los procesos de cara a las consolas, posicionamiento de los procesos en curso en los segmentos de memoria, etc...

2.5.2.7. TOD

TOD (Time Of Date) permite inicializar y visualizar la fecha del día (año, mes, día, hora, minuto, segundo). Este programa de utilidad posee un calendario universal que permite visualizar el nombre del día.

Ejemplo:

```
OA>TOD 05/17/82 10:21:00          17 Mayo 1982 10h 21 min
Strike key to set time <cr>
Mon 05/17/82 10:21:00
OA>TOD
Mon 05/17/82 10:22:38
```

2.5.2.8. ABORT

Este mandato da al usuario la posibilidad de abortar la ejecución del programa que da como argumento.

Ejemplo:

```
OA>ABORT PROGTEST
```

2.5.2.9. SPOOL

Este mandato permite enviar un fichero texto en ASCII al periférico asignado a la impresora, incluso aunque esté ocupada. El fichero a imprimir se coloca en un fichero de espera que se vacía automáticamente en cuanto la impresora queda libre.

Ejemplo:

```
OA>SPOOL B:EDITION1.PRN,EDITION2.TEXT,EDITION3.DOC
```

El mandato STOPSPLR permite abortar la salida por la impresora.

2.5.2.10. SCHED

Este mandato permite solicitar la ejecución de un programa a partir de una fecha y hora determinada, definida por el usuario:

```
OA>SCHED 5/17/85   11:15 MIDIA
```

El programa MIDIA se ejecutará el 17 de mayo de 1985 a las 11h 15 m.

2.5.2.11. PRLCOM

Este mandato permite transformar un fichero ejecutable de tipo PRL en fichero ejecutable de tipo COM.

Ejemplo:

```
OA>PRLCOM PROGMPM.PRL B:PROGCPM.COM
```

2.5.2.12. GENHEX

Este mandato permite reconstruir un fichero objeto en formato Intel hexadecimal de tipo HEX a partir de un fichero ejecutable de tipo COM. Se puede dar un desplazamiento como parámetro para especificar la posición de comienzo.

Ejemplo:

```
OA>GENHEX PROGRAM. COM 100
```

2.5.2.13. GENMOD

Este mandato permite generar un fichero ejecutable de tipo PRL a partir de un fichero constituido por la concatenación de 2 imágenes de un mismo objeto de tipo HEX desfasadas la una en relación a la otra 0100H.

Ejemplo:

```
OA>GENMOD CONCATEN.HEX B:RESULT.PRL
```

Los ficheros de tipo PRL son ficheros ejecutables en un segmento de memoria, al contrario de lo que ocurre con los ficheros de tipo COM que son ejecutables en el área TPA.

La concatenación de dos ficheros desfasados en 100H permite determinar los campos de dirección del programa y renombrarlos en un "MAP de bits" contiguo al objeto binario. A cada octeto binario generado se le asocia un bit que, si es un 1, indicará al cargador del programa que hay que trasladar la dirección (peso fuerte) un valor igual a la dirección de comienzo del segmento asignado.

También se reserva un espacio de 256 octetos en la cabecera de todos los ficheros PRL en donde se memoriza el tamaño del programa.

Ejemplo:

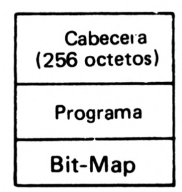


Fig. 12.– Estructura de un fichero PRL.

2.5.2.14. GENSYS

GENSYS configura el sistema conforme al entorno físico (hardware) y de acuerdo con el criterio del usuario: número de consolas, memoria con o sin bancos, número de la interrupción utilizada por el DDT para los "breakpoints", nombres de los procesos residentes de tipo RSP, etc... El ejemplo que damos a continuación nos muestra el desarrollo interactivo de la generación:

```
OA>GENSYS
MP/M System Generation

Top page of memory = FF                     tamaño máximo en páginas
Number of consoles = 2                      dos consolas
Breakpoint RST # = 6                        número de RST para DDT
Add system call user stacks  (Y/N)? Y
Z 80 CPU (Y/N)? Y
```

```
Bank switched memory (Y/N)? Y                bancos memoria
Banked BDOS file manager (Y/N) N
Enter memory segment table: (ff terminates list)
Base,size,attrib,bank = 0,BC,0,0             segmento 0 = BC00H octetos
Base,size,attrib,bank = 0,C0,0,1             segmento 1 = C000H octetos
Base,size,attrib,bank = FF                   fin de descripción
Select Resident System Processes: (Y/N) Y
SPOOL      ?N
TIME       ?Y
OA>
```

2.5.2.15. MPMLDR

MPMLDR carga en memoria el fichero "MPM.SYS" después de que se ha hecho la generación del sistema con SYSGEN. MPMLDR sólo puede ejecutarse bajo MP/M o mediante la operativa de "bootstrap" (cold start loader) del MP/M. La carga de la configuración del sistema queda visualizada de la siguiente forma:

```
MP/M Loader

Number of consoles     = 2
Breakpoint RST #       = 6
Z 80 CPU
TOP of memory          = FFFFH
Memory Segment Tables:

SYSTEM     DAT     FF00H     0100H
CONSOLE    DAT     FD00H     0200H
USERSYS    STK     FC00H     0100H
XIOS       SPR     F600H     0600H
BDOS       SPR     E200H     1400H
XDOS       SPR     C300H     1F00H
TIME       RSP     C600H     0300H

Memseg     Usr     0000H     C000H     Bank 01H
Memseg     Usr     0000H     BC00H     Bank 00H

MP/M
OA>
```

2.6. SERVICIOS DE SISTEMA OFRECIDOS

Al igual que los mandatos, se han conservado las funciones del sistema ofrecidas por el BDOS del CP/M, con algunas excepciones, y se han añadido nuevas funciones específicas del MP/M.

2.6.1. Funciones suplementarias del XDOS

- Solicitud y liberación de espacio de memoria absoluto o reubicable.
- Consulta de los periféricos (modalidad "polling")
- Espera y activación de sucesos.
- Creación, apertura y supresión de una cola.
- Lectura y escritura, condicional o no, de una cola
- Puesta en espera durante un tiempo (retardo).
- Llamada al "dispatcher" de procesos.
- Creación y terminación de un proceso.
- Inicialización de la prioridad.
- Envío de un mandato
- Análisis sintáctico de un nombre de fichero.
- Obtención del número de consola.
- Inicialización y restauración de la fecha.
- Abandono de un proceso específico.

2.6.2. Primitivas de sistema del XIOS

El XIOS es una extensión del BIOS del CP/M; nos encontramos con las mismas primitivas. Se han añadido unas primitivas específicas del contexto MP/M. Los puntos de entrada se encuentran a continuación del vector de bifurcación del BIOS.

```
XIOS+51   JMP SELMEMORY    ; selección de un banco de memoria
XIOS+54   JMP POLLDEVICE   ; consulta de un periférico
XIOS+57   JMP STARTCLOCK   ; arranque del reloj
XIOS+60   JMP STOPCLOCK    ; parada del reloj
XIOS+63   JMP EXITREGION   ; salida de una fase crítica
XIOS+66   JMP MAXCONSOLE   ; número máximo de consolas
XIOS+69   JMP SYSTEMINIT   ; inicialización del sistema
XIOS+72   JMP IDLE         ; puesta en estado de reposo
```

3

Extensiones de la familia CP/M

3.1. SISTEMAS PARA MICROPROCESADORES 16 BITS CP/M-86, MP/M-86

La aparición en el mercado, a principios de los años 80, de los microprocesadores de 16 bits ha llevado a Digital Research a desarrollar los sistemas operativos CP/M-86 y MP/M-86 sobre el 8086 de Intel. Una de las mejoras que aporta el 8086 es la capacidad de memoria, que puede pasar de un Mega-octeto.

3.1.1. CP/M-86

Se han conservado casi todas las facilidades del CP/M-80 y se han introducido algunas mejoras. De una manera general, el CP/M-86 mantiene la compatibilidad a nivel de ficheros, con todas las versiones anteriores del CP/M. Siendo diferente el microprocesador, no se puede asegurar esta compatibilidad a nivel de código máquina y de gestión de memoria. Así pues, aunque los binarios sean distintos, existen unos programas de utilidad que permiten el cambio de código de un fichero ejecutable 8080 a un fichero ejecutable 8086.

3.1.1.1. Arquitectura del CP/M-86

Al contrario que el CP/M, el CP/M-86 no reside en las dos primeras pistas reservadas, sino en un fichero llamado "CPM.SYS". En efecto, como ya no hay problemas de memoria, el CP/M-86 es demasiado grande como para caber en estas dos pistas. En ellas está el "Cold Start

Loader" LDBIOS cuya misión es la de cargar el fichero CPM.SYS en memoria.

En cuanto a la estructura interna del CP/M-86, nos encontramos con los mismos módulos de que consta el CP/M:

- el CCP: interfase usuario-máquina
- el BDOS: gestión de los ficheros y funciones del sistema
- el BIOS: núcleo físico ligado al entorno hardware.

El CCP y el BDOS ocupan cerca de 10 K-octetos, mientras que el tamaño del BIOS varía dependiendo de las instalaciones. El CCP no se puede solapar con programas en el área TPA, lo cual hace que el CP/M 86 sea totalmente residente. Este no es el caso del CP/M.

3.1.1.2. Los mandatos del CP/M-86

El código binario, en relación al CP/M, es diferente y los ficheros ejecutables vienen identificados por el tipo "CMD". Como bajo CP/M, el promp "A>" es el mensaje de invitación para que el usuario teclee un mandato.

Los mandatos internos del CCP, como:

DIR ERA REN TYPE USER

ofrecen la misma funcionalidad que bajo CP/M. Así mismo los programas de utilidad no residentes, como:

STAT PIP ED SUBMIT

operan de idéntica forma.

Por el contrario, los productos ASM86 y DDT86, aunque mantienen la misma filosofía que los productos ASM y DDT, están francamente modificados.

ASM86 lee una fuente en ensamblador 8086 de tipo A86 y produce tres ficheros: un objeto de tipo H86 en formato Intel, un listado de tipo LST y una tabla de símbolos de tipo SYM.

La sintaxis del mandato ASM86 acepta argumentos si se incluye el carácter "$". Estos argumentos están formados por dos letras: la primera representa el tipo de fichero (A = fuente, H = objeto, P = listado, S = símbolo) y la segunda el nombre del disco asociado.

Ejemplo de llamada:

```
A>ASM86 MODULE $AB HB PC SB
```

se ensambla el programa "MODULE.A86" que se encuentra en el disco B y se obtiene un objeto de tipo H86 en el disco B, un listado de tipo LST en el disco CP/M y la tabla de símbolos en el disco B.

El mandato GENCMD (Generate CMD) sustituye al mandato LOAD del CP/M, creando un fichero ejecutable de tipo CMD a partir de un fichero objeto en formato Intel de tipo H86.

El mandato LDCOPY (loader Copy) sustituye al mandato SYSGEN.

Los mandatos ASM86 y GENCMD traen también incorporado el tipo COM para permitir el desarrollo cruzado bajo CP/M-80.

3.1.2. MP/M-86

El sistema MP/M-86 constituye la gama superior de la familia de los sistemas operativos de Digital Research. En efecto, reune las funcionalidades específicas del microprocesador 8086 de 16 bits, obtenidas del CP/M-86, y las del MP/M (multi-puestos, multi-tareas, multi-usuarios).

3.1.2.1. Arquitectura del MP/M-86

Como el MP/M, el MP/M-86 está organizado en varios módulos:

- el TMP: interfase de diálogo con los usuarios
- el SUP: supervisor y tratamiento de las llamadas al sistema
- el RTM: distribuidor de procesos y gestión de las colas
- el MEM: módulo de gestión de la memoria
- el CIO: módulo de gestión de entradas-salidas de caracteres
- el BDOS: sistema de gestión de ficheros
- el XIOS: núcleo físico del tratamiento de las entradas-salidas

Observamos que el antiguo XDOS del MP/M ha sido desglosado en cuatro módulos independientes (SUP, RTM, CIO y BDOS) bajo MP/M-86.

SUP El supervisor gestiona las interacciones entre los procesos de los usuarios y los otros módulos del sistema. Todas las llamadas al sistema realizadas por los usuarios o por los módulos internos pasan por el supervisor que hace las funciones de controlador. El intérprete de mandatos CLI (se le puede llamar desde cualquier nivel) también forma parte del supervisor.

RTM El monitor de tiempo real RTM es el núcleo de la "multitarea" del sistema. Permite gestionar la conmutación y la asignación de la CPU a los procesos. Además de la distribución de los procesos, el RTM se encarga de la gestión de las colas, la gestión de los sucesos lógicos, la consulta cíclica a los periféricos (polling) y la gestión de las bases de tiempos.

CIO Este módulo gestiona las entradas-salidas en modo carácter para las consolas o las impresoras. A cada periférico se le asocia un CCB (bloque de control de carácter) que contiene las informaciones relativas al propietario, al periférico y a los caracteres de edición.

BDOS Este módulo representa el sistema clásico de gestión de ficheros ya conocido bajo otros sistemas. Ofrece al usuario todas las funciones precisas para la gestión de ficheros, y gestiona la asignación del espacio de disco. En relación al MP/M, se han introducido algunas mejoras para solucionar los conflictos de acceso a nivel de ficheros (bloqueos, compartición del acceso, reserva del elemento, etc...).

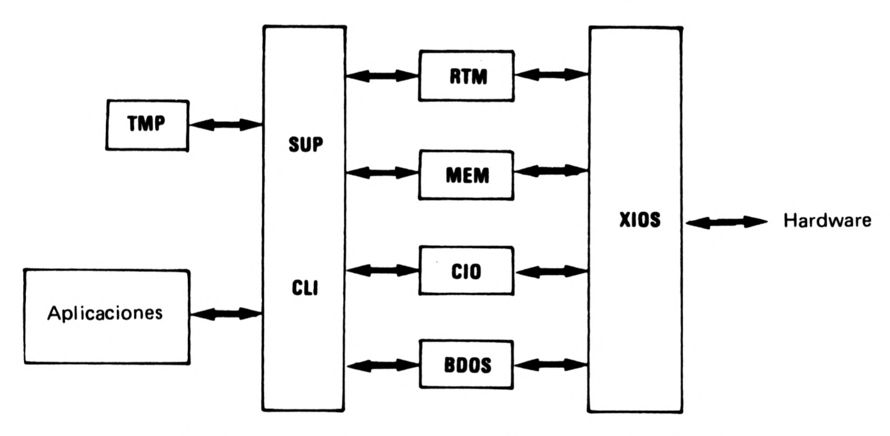


Fig. 13.— Arquitectura de los módulos de MP/M-86.

3.1.2.2. Los mandatos del MP/M-86

El "prompt" de invitación para teclear un mandato es el mismo que bajo MP/M, al igual que la sintaxis del mandato. El MP/M-86 incluye, además, el concepto de protección individual a nivel de un fichero. Se

puede especificar una palabra clave "password" en una referencia de fichero, con la siguiente sintaxis:

d: nombre del fichero.tipo; palabra clave

Ejemplo:

A:DOCUMENT.DST;SECRET

Incluso a los propios mandatos, que son ficheros ejecutables de tipo CMD (ver CP/M-86), se les puede asignar una palabra clave, si lo que se desea es que no sean públicos (mandatos privilegiados).

Ejemplos:

0A>GENSYS;SESAM

4C>B:ESPION;SPY

Los mandatos del MP/M-86 reagrupan a la mayoría de mandatos del MP/M y el CP/M-86. Se han incluido cuatro nuevos mandatos:

PRINTER permite visualizar o cambiar los números de las impresoras conectadas al sistema.

SDIR permite visualizar los directorios de los ficheros (directory) con numerosas opciones.

SET permite modificar los niveles de protección (palabra clave), la retención y los atributos de un fichero.

SHOW permite visualizar el estado de un disco y sus protecciones.

3.2. SISTEMAS ORIENTADOS A REDES: CP/NET, CP/NOS, MP/NET

CP/NET, CP/NOS y MP/NET, desarrollados por Digital Research, son unos sistemas operativos orientados a "redes". Permiten interco-

municar varios micro-ordenadores basados en el CP/M y el MP/M, acceder a recursos comunes y compartir los dispositivos de entradas-salidas.

3.2.1. CP/NET

El CP/NET es el primer sistema de la familia "red". Permite interconectar dos micro-ordenadores, de los cuales uno funciona bajo MP/M (sistema maestro) y el otro bajo CP/M (sistema esclavo). El sistema maestro lleva la iniciativa del diálogo y gestiona los recursos a compartir. Así pues, el acceso a recursos comunes es factible partiendo de dos sistemas, maestro y esclavo.



3.2.2. CP/NOS

CP/NOS es un sistema que permite la comunicación entre un sistema MP/M maestro y un sistema esclavo CP/M en PROM y RAM reducido a la mínima expresión, es decir, un CP/M sin soporte en disco. Sólo se soportan la gestión de la consola y de la impresora.

3.2.3. MP/NET

El MP/NET representa la gama superior de la familia "redes". Permite la interconexión de varios sistemas MP/M. Ya no existe la distinción de sistema maestro y sistema esclavo, con lo cual la red es totalmente simétrica. El conjunto da lugar a un ambiente "multi-micro-ordenadores".



3.2.4. Arquitectura del sistema CP/NET

El sistema esclavo CP/M del CP/NET está lógicamente dividido en cuatro módulos.

- BIOS: idéntico al del CP/M
- BDOS: idéntico al del CP/M
- SNIOS: (Slave Network I/I System) extensión red del BIOS
- NDOS: (Network Disk Operating System) extensión red del BDOS

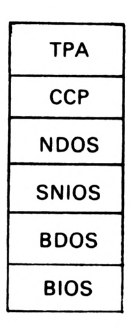


Fig. 14.— Implantación en memoria de los módulos de CP/NET esclavo.

El sistema maestro MP/M del CP/NET se construye sobre un MP/M que posee dos módulos suplementarios: un módulo lógico SLVSP residente de tipo RSP que gestiona las funciones lógicas de intercambio de mensajes con el sistema esclavo y el módulo NETWRKIF que contiene los sub-programas de gestión de entradas-salidas específicas del ambiente "red". Al usuario se le facilita este último módulo, como el módulo SNIOS del sistema esclavo, como un "fuente" para el caso en que se deseen añadir modificaciones ligadas al hardware o al procedimiento de la comunicación.

3.2.5. Los mandatos CP/NET

El CP/NET incluye, en relación al CP/M y al MP/M, una serie de nuevos mandatos específicos de la "red". Los detallamos a continuación.

LOGIN permite a un usuario conectarse a la red para acceder a los recursos comunes. La palabra clave es opcional.

A > LOGIN <palabra clave>

LOGOFF desconecta a un usuario de un sistema esclavo del sistema maestro.

```
A>LOGOFF
```

SNDMAIL permite enviar un mensaje desde un sistema esclavo hacia un sistema esclavo o maestro.

```
A>SNDMAIL <destinatario> "mensaje a enviar"
```

RCVMAIL permite obtener todos los mensajes enviados por un sistema maestro.

```
A>RCVMAIL
```

NETWORK permite a un usuario esclavo actualizar la tabla de configuración de los recursos comunes del sistema esclavo.

```
A>NETWORK >perif.local> = <perif. maestro>
```

LOCAL permite reasignar al sistema local los periféricos ya asignados a la red.

```
A>LOCAL <perif.local>
```

DSKRESET es idéntico al del MP/M en el sistema local.

```
A>DSKRESET
A>DSKRESET A:,B:
```

ENDLIST permite enviar un fin de fichero <ctrl-Z> al periférico asignado a la impresora.

```
A>ENDLIST
```

CPNETLDR permite cargar en memoria los módulos del sistema CP/NET: SNIOS.SPR y NDOS.SPR.

A>CPNETLDR

CPNETSTS — visualiza la tabla de configuración del sistema esclavo.

A>CPNETSTS

BROADCST — permite a un usuario de un sistema maestro enviar un mensaje a todos los otros usuarios.

OA>BROADCST "mensaje a difundir sobre todos"

MSNDMAIL — permite a un usuario de un sistema maestro enviar un mensaje hacia un sistema esclavo.

3O>MSNDMAIL <destinatario> "mensaje"

MRCVMAIL — permite obtener todos los mensajes enviados por los sistemas esclavos.

2B>MRCVMAIL

SPOOL — permite a los usuarios utilizar la impresora del sistema maestro.

1D > SPOOL fichero1, fichero2,...

4

Productos desarrollados alrededor del CP/M y MP/M

4.1. EL FENOMENO CP/M

Paralelamente al desarrollo del CP/M y de sus extensiones, son muchas las empresas que han producido un software compatible con el CP/M. Lo que ha hecho del CP/M un verdadero fenómeno, y de hecho, un auténtico Stándar, es la existencia de una amplia biblioteca de programas. Estos productos, como macro-ensambladores, intérpretes, compiladores, editores de documentos, utilidades, etc., no se suministran con el CP/M, pero están disponibles como opciones o se les puede comprar en los supermercados de software.

4.2. MACRO-ENSAMBLADORES

4.2.1. MAC

Además del ensamblador ASM que forma parte del "paquete" (package) CP/M, Digital Research ha desarrollado el macro-ensamblador MAC que tiene numerosas similitudes con el macro-ensamblador ASM-80 de Intel.

La salida que produce el MAC son los ficheros clásicos de tipo PRN (impresora) y HEX (objeto), así como el fichero que contiene la tabla de símbolos de tipo SYM utilizable por el "debugger" SID.

El paquete MAC incluye, además, un conjunto de macro-instrucciones en el fichero SEQIO.LIB que simplifica la interfase entre el programa ensamblador y los servicios ofrecidos por el sistema. Otro fichero llamado Z80.LIB da al MAC la posibilidad de ensamblar las instrucciones del Z80. Ejemplo de puesta en funcionamiento:

```
A>MAC B:UTIL
```

4.2.2. MACRO-80

El macro-ensamblador MACRO-80 ó M80 de Microsoft satisface igualmente las especificaciones del macro-ensamblador ASM-80 de Intel. MACRO-80 permite ensamblar programas escritos en 8080 o en Z80 y produce un fichero objeto de tipo REL (objeto reubicable) el cual, una vez tratado por LINK-80, da lugar a un fichero ejecutable de tipo COM.

Los otros lenguajes de Microsoft (BASIC-80, FORTRAN-80 y COBOL-80) pueden relacionarse fácilmente, con ayuda de las referencias globales, con los sub-programas escritos en MACRO-80.

El editor de enlaces LINK-80, la utilidad de referencias cruzadas CREF-80 y el gestor de librerías LIB-80 forman parte integrante del "paquete" MACRO-80 y se suministran conjuntamente.

La llamada al macro-ensamblador se puede formular de dos maneras

```
M80
```

ó

```
M80 fichero-objeto, fichero-listado=fichero-fuente
```

En el primer caso se envía el prompt "*" al usuario para que especifique sus ficheros tal y como se indican en el segundo formato. Sólo son obligatorios el signo igual y el nombre del fichero fuente que va a continuación; el nombre del fichero objeto toma entonces el tipo REL, por defecto, y el tipo PRN queda asociado al listado. Ejemplos:

```
A>M80
*OBJET,LISTING=SOURCE        ensambla SOURCE.MAC
                             produce LISTING.PRN y OBJET.REL
A>M80 PROG,PROG=PROG         ensambla PROG.MAC
                             produce PROG.PRN y PROG.REL
```

MACRO-80 también acepta las opciones al final de la cadena de mandato, precedidas del carácter "/".

O representación octal del código generado
H representación hexadecimal del código generado
R fuerza la generación de un fichero objeto de tipo REL
L fuerza la generación de un fichero listing de tipo PRN
C fuerza la generación de un fichero de referencias cruzadas
Z ensamblaje de un programa escrito en Z80.

Ejemplo:

A>M80 PROG/R/L/Z   el fuente está en el ensamblador Z80 (Z)
se obtiene PROG.REL (R) y PROG.PRN (L)

4.3. INTERPRETES

4.3.1. MBASIC

El intérprete BASIC versión 5 de Microsoft, llamado MBASIC, se ha convertido en un verdadero estándar. Al contrario de lo que sucede con otros intérpretes, MBASIC traduce cada línea fuente en código binario intermedio "sobre la marcha" lo cual hace que el tiempo de traducción sea prácticamente imperceptible.

El código intermedio, extremadamente compacto se interpreta cuando el usuario teclea el mandato RUN. Además es muy fácil y rápido el desarrollo de los programas. MBASIC lleva incorporado un editor y permite la parada de la ejecución del programa en cualquier momento para visualizar o cambiar el valor de las variables y reanudar después la ejecución. Estas son las principales mejoras aportadas por el MBASIC:

- nombres de variables más largos (hasta 40 caracteres)
- tipos de variables: INTEGER, REAL, DOUBLE PRECISION, STRING
- programación estructurada de los bucles: instrucciones WHILE/WEND
- numeración y renumeración automática de las líneas AUTO/RNUM)

– representación octal, hexadecimal y binaria de la información
– varias instrucciones por línea, separadas por los ":"
– seguimiento de los números de línea durante la ejecución (mandatos TRON/TROFF).

Ejemplos:

A>MBASIC	llamada del BASIC
LOAD "B:PROGBAS"	carga del programa en memoria
RUN	inicio de la ejecución
SAVE "PROBASIC",A	salva el fuente en ASCII
SYSTEM	retorno al sistema
A>	

4.3.2. CBASIC

CBASIC de Compiler Systems es un intérprete BASIC no interactivo. Se necesitan tres fases para ejecutar un programa:

– la construcción del programa por un editor de textos independiente.
– la traducción del fuente en un fichero intermedio de tipo INT;
– la ejecución propiamente dicha por interpretación del código intermedio.

El mandato CBASIC realiza la traducción y el mandato CRUN inicia la interpretación del fichero INT que contiene el código intermedio.

Una de las características de CBASIC es que no son necesarios los números de líneas.

Sintaxis de puesta en funcionamiento:

CBASIC fuente	compila el programa "fuente.BAS" y produce un fichero de tipo INT
CRUN objeto	interpreta el fichero de tipo INT

4.3.3. PASCAL/M

PASCAL/M de Sorcim es un intérprete de código-P PASCAL. El código-P es un lenguaje intermedio independiente de la máquina.

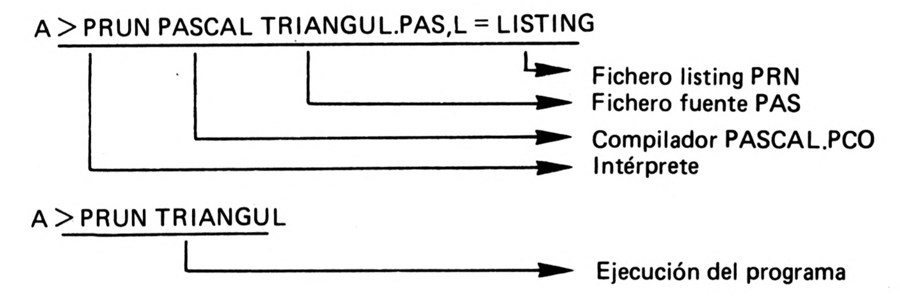
PASCAL/M produce tres ficheros:

- PASCAL.PCO: compilador PASCAL bajo forma de código-P
- PASLIB.PCO: librería PASCAL (Run-time) bajo forma de código-P
- PRUN.COM: intérprete PASCAL de código-P

La sintaxis del PASCAL/M es:

PRUN PASCAL fuente	interpreta el compilador PASCAL y traduce el fuente PAS en código-P PCO
PRUN objeto (código-P)	interpreta el programa usuario traducido en código-P

Ejemplo:



4.3.4. CIS-COBOL

CIS-COBOL, siguiendo la norma ANSI-74, produce, a partir de un fuente de tipo COB, un código intermedio compactado, de tipo INT, ejecutable por el intérprete RUN. Este compilador está fuertemente orientado al tratamiento conversacional disponiendo de numerosos dispositivos para la gestión de la pantalla, como el posicionamiento del cursor y los verbos ACCEPT y DISPLAY. CIS-COBOL también permite el tratamiento de ficheros secuenciales, directos e indexados y el solapamiento automático de programas.

Sintaxis de puesta en funcionamiento:

COBOL fuente	compilación de un fuente de tipo COB.
RUN objeto.INT	interpretación del programa compilado

4.3.5. MU-LISP

MU-LISP de Microsoft es un intérprete de tratamiento de listas orientado a la "inteligencia artificial". Contiene 83 funciones LISP y puede trabajar con números con precisión infinita.

4.4. COMPILADORES

4.4.1. BASCOM

El compilador BASCOM de Microsoft es la prolongación del intérprete MBASIC, enteramente compatible con él en lo que al lenguaje se refiere.

BASCOM produce, a partir de un programa fuente BASIC de tipo BAS, un fichero intermedio reubicable de tipo REL, que tratado por el editor de enlaces LINK-80 proporciona el código máquina 8080 ó Z80 en un fichero tipo COM.

Se ha perdido el carácter interactivo de la puesta a punto que existía bajo MBASIC en beneficio de la velocidad de ejecución del programa compilado por BASCOM. Así pues, recomendamos desarrollar un programa con MBASIC, y una vez puesto a punto, compilarlo con BASCOM para su ejecución.

Sintaxis de puesta en funcionamiento:

```
BASCOM objeto, listado = fuente/opciones
```

Sólo nos queda añadir que la sintaxis del MACRO-80 es enteramente compatible con la del BASCOM. Ejemplo:

```
A>BASCOM PROGBIN,PROGLIS=PROGBAS
```

4.4.2. SBASIC

SBASIC es un verdadero compilador estructurado que genera código 8080 en un fichero tipo HEX. Se han incorporado las instrucciones de control del PASCAL WHILE...DO, IF...THEN...ELSE, BEGIN...END. Las "funciones" y los "procedimientos" constituyen otra característica del SBASIC.

4.4.3. PASCAL/MT+

El PASCAL/MT+ procede de la combinación entre el PASCAL estándar (norma ISO), del PASCAL/MT (subconjunto del PASCAL estándar) y de un programa simbólico para la puesta a punto. Acepta las variables reales en coma flotante o en BCD. El lenguaje de ensamblaje, así como las entradas-salidas y los procedimientos de interrupción están interfasados.

PASCAL/MT+ es un compilador que produce un módulo reubicable de tipo ERL compatible con el editor de líneas LINKMIT.

Ejemplo:

```
A > MTPLUS  CRIBLE,PAS
A > LINKMT  CRIBLE,PASLIB/S
```

4.4.4. PASCAL/Z

El PASCAL/Z de Ithaca Intersystems es un compilador que sólo produce código máquina Z80. De hecho, la compilación se realiza en dos etapas: la traducción del fuente en código mnemónico ensamblador Z80 en un fichero tipo SRC, y el ensamblaje propiamente dicho que produce el binario Z80 en un fichero en formato Intel de tipo HEX.

Sintaxis de puesta en funcionamiento:

```
PASCALZ fuente.opciones
```

donde "fuente" representa el nombre del fichero PASCAL de tipo PAS, y las "opciones" una lista opcional de tres letras (nombres de los discos para los ficheros tipo PAS, SRC y LST).

```
ASMBL fuente1.opt1, fuente2.opt2,...
```

donde los "fuentes" representan los nombres de los ficheros traducidos en ensamblador Z80 de tipo SRC, y "optn" la lista opcional de los nombres de los discos que contienen los ficheros tipo SRC, HEX, LST.

Ejemplo:

```
A>PASCALZ PROPASCA          compilación propiamente dicha
```

A>ASMBL LIBS,PROPASCA	ensamblaje del programa PROPASCA.SRC con la librería PASCAL: LIBS.SRC

4.4.5. FORTRAN-80

El FORTRAN-80 de Microsoft satisface la norma ANSI-66 excepto para los "complejos". FORTRAN-80 se distribuye con su librería FORLIB.REL así como con el LINK-80 y los productos que le acompañan.

Sintaxis de puesta en funcionamiento:

```
F80 objeto, listado = fuente
```

El fuente debe ser de tipo FOR y el fichero objeto obtenido es de tipo REL. A continuación se edita este objeto con LINK-80 y se genera un fichero ejecutable de tipo COM.

Ejemplo:

```
A>F80 CALCUL,LST:=CALCUL
```

4.4.6. COBOL-80

COBOL-80 satisface la norma ANSI-74, y las especificaciones de Microsoft: fuente de tipo COB y obtención de un objeto reubicable de tipo REL. COBOL-80 permite el tratamiento de los ficheros secuenciales, relativos e indexados. Se facilita la gestión interactiva de la pantalla con los verbos ACCEPT y DISPLAY que permiten la transferencia de páginas completas. Se permite el particionado de aquellos programas que sobrepasan el tamaño de la memoria disponible.

Dado que este compilador es muy voluminoso, está segmentado en varias partes:

COBOL.COM	programa principal del compilador residente
COBOLn.OVR	cuatro overlays desde n= 1 a n= 4
COBLIB.REL	librería "Runtime"
CRTDRV.REL	"driver" para terminal vídeo

La compilación se realiza en cinco fases. "COBOL.COM" compila la IDENTIFICATION DIVISION y la ENVIRONMENT DIVISION.

"COBOL1.OVR" compila la DATA DIVISION y "COBOL2.OVR" la PROCEDURE DIVISION. Estas tres fases constituyen el primer paso de la compilación, creándose un fichero traducido en código intermedio "STEXT.INT". "COBOL3.OVR" traduce entonces el fichero intermedio en un fichero reubicable de tipo REL. Por último "COBOL4. OVR" asigna los FCBs necesarios y destruye el fichero temporal.

Sintaxis de puesta en funcionamiento:

```
COBOL objeto, listado = fuente
```

Ejemplo:

```
A>COBOL GESTION,TTY:=GESTION
```

4.4.7. C

El lenguaje C inicialmente desarrollado por los Laboratorios Bell para el sistema operativo UNIX, es un lenguaje de alto nivel cuya estructura recuerda la del PASCAL, sin los tipos. Este lenguaje, orientado al "desarrollo de software", manipula los punteros, las estructuras, permite la inclusión de los segmentos "fuente" y las compilaciones "separadas". La formulación de las expresiones le permite generar un código objeto mucho más compacto y, por ejemplo, se pueden introducir ciertas variables en los registros de la CPU.

Existen varios compiladores C disponibles bajo CP/M: el "C" de Whitesmiths, el "C" de BDS. el "Small C", el "Tiny C", etc...

El CP/M de Whitesmith es la herramienta más potente en materia de "Lenguaje C". Dispone de más de 75 funciones para gestionar las entradas-salidas, las cadenas de caracteres y la asignación de la memoria. El código producido es de tipo REL compatible con el editor de enlaces LINK-80. El compilador se ejecuta a partir de una llamada a SUBMIT con un mandato catalogado de tipo SUB. El fichero resultado es de tipo COM y después de la edición de los enlaces es directamente ejecutable.

4.5. EDITOR DE ENLACES LINK-80

El editor de enlaces permite entrelazar módulos objetos de tipo REL, y construir un único fichero ejecutable de tipo COM. De hecho,

los objetos de tipo REL son independientes del lenguaje máquina y constituyen un código objeto intermedio específico de los compiladores de Microsoft: BASIC-80, FORTRAN-80, COBOL-80 y MACRO-80. El LINK-80 se encarga de interpretar este código y de general el código máquina asociado.

La sintaxis de llamada al editor de enlaces puede ser de dos formas:

```
L80
```

ó

```
L80 objeto1, objeto2, ..., objeton
```

En el primer caso, se envía un prompt "*" a la consola para que el usuario teclee a continuación los nombres de los ficheros objeto tal y como se definen en el segundo formato.

Detrás de cada nombre de fichero se pueden añadir algunos parámetros suplementarios precedidos de un "/".

- E Generación del código binario en memoria y retorno al sistema
- F Solicitud de ejecución del programa después de la edición de enlaces
- M Edición de la tabla de referencias globales (librerías)
- N Se salva el nombre del fichero ejecutable con el tipo COM

Ejemplo:

```
A>L80 PROGMAC,PROGCOB,B:PROGBAS,PROGRAM/M/N/E
```

4.6. TRATAMIENTO DE TEXTO

4.6.1. WORDSTAR

El paquete (Software) WORDSTAR de Micropro es un editor orientado al "tratamiento de textos" que combina, a la vez, la edición de texto clásico y la edición de documentos. WORDSTAR es un editor de "página entera" que necesita una pantalla de vídeo con gestión del cursor en X e Y.

Este editor visualiza directamente el documento en pantalla en el momento que lo solicite el usuario. Las líneas de texto quedan ajustadas automáticamente al añadir las secuencias <cr> <lf> en el lugar adecuado e insertando espacios suplementarios repartidos equitativamente entre las palabras de la línea. El usuario puede, en cualquier momento, definir los parámetros de "formateo" del texto: determinación de los márgenes de izquierda y derecha, longitud de las páginas, cabeceras y pies de página, caracteres en negrita, subrayados, etc...

La primera mitad de la pantalla se reserva a la explicación de cada uno de los mandatos. El usuario puede seleccionar hasta cuatro niveles de ayuda, en forma de menús.

Un programa independiente, llamado "INSTALL" permite configurar el WORDSTAR para un terminal dado. INSTALL posee una lista de varias decenas de terminales con sus características. Basta seleccionar de la lista el número correspondiente al terminal de su sistema. Si no existiese, INSTALL le ofrece la posibilidad de definir por Vd. mismo las características del terminal: deberá hacer referencia a las notas explicativas del producto. Ejemplo de llamada:

```
    A>WS
o
    A>WS B:RAPPORT.DOC
```

4.6.2. WORDMASTER

El WORDMASTER constituye un subconjunto del WORDSTAR en el que se ha eliminado toda la parte de edición de documentos. WORDMASTER es pues, un editor de texto de "página completa" que contiene todos los mandatos del editor estándar ED del CP/M.

4.6.3. TEXTWRITER III

Este producto es un "formateador" de texto para encuadrar o paginar documentos. En el documento sobre el que se está trabajando se pueden insertar otros documentos, ya preparados y almacenados en disco (cartas preimpresas, "correo", contratos,...)

4.7. PAQUETES DE GESTION

4.7.1. DATASTAR

DATASTAR de Micropro es un paquete de gestión para la captura, consulta y actualización de ficheros. Es un producto orientado a la "pantalla" que permite al usuario definir sus formatos de captura de datos. Estas pueden constar de varias páginas de longitud y hasta un máximo de tres páginas de anchura. Unas máscaras de captura, permiten la protección y el control de los campos (tamaño, caracteres numéricos o alfabéticos,...). En pantalla se visualiza un menú de ayuda al usuario, con el conjunto de mandatos disponibles.

4.7.2. SUPERSORT

SUPERSORT es un paquete de clasificación, fusión y "formateado" de ficheros. Permite clasificar o fusionar hasta 32 ficheros de entrada, ya sean de tipo ASCII, BCD o binario. Los registros pueden ser de longitud fija o variable hasta un máximo de 2048 caracteres. Las claves de las clasificaciones son independientes, y su número puede variar desde 1 a 32.

4.7.3. SUPERCALC

El paquete SUPERCALC de Sorcim es similar al producto VISICALC, muy conocido en Apple-2. SUPERCALC es un paquete orientado a la gestión financiera, la previsión de las tendencias y la ayuda a la toma de decisiones.

De hecho, es una gigantesca matriz (parrilla) de cálculo que utiliza una pantalla. Los datos se localizan en celdas referenciadas por una letra para las columnas y un número para las filas. Estas celdas pueden contener valores numéricos, cadenas de caracteres alfanuméricos, o fórmulas. Cada modificación de un valor provoca el ajuste y el cálculo automático, allí donde aparece dicho valor, en todas las celdas del modelo.

SUPERCALC permite gestionar hasta 63 columnas (A-Z, AA-AZ, BA-BK) y 254 filas (1-254). La matriz de celdas puede desplazarse horizontal y verticalmente en pantalla; el usuario ve una "ventana" de la matriz.

4.7.4. DBASE II

DBASE-II es un sistema de gestión de base de datos (SGBD) interactivo relacional y no jerárquico. Posee su propio lenguaje de tratamiento de estructuras. Una serie de sencillos mandatos permiten la modificación, cambio o ampliación de las bases de datos.

Las especificaciones principales de DBASE-II soportan:

- 65536 registros como máximo por fichero de base de datos.
- 32 campos de datos como máximo por registro.
- 1000 caracteres como máximo por registro.
- 254 caracteres como máximo por cada campo.
- 10 cifras de precisión numérica.

ANEXOS

Anexo A

Tabla de códigos ASCII

	0x	1x	2x	3x	4x	5x	6x	7x
x0	NUL	DLE	SP	0	@	P	`	p
x1	SOH	DC1	!	1	A	Q	a	q
x2	STX	DC2	"	2	B	R	b	r
x3	ETX	DC3	#	3	C	S	c	s
x4	EOT	DC4	$	4	D	T	d	t
x5	ENQ	NAK	%	5	E	U	e	u
x6	ACK	SYN	&	6	F	V	f	v
x7	BEL	ETB	'	7	G	W	g	w
x8	BS	CAN	(	8	H	X	h	x
x9	HT	EM	)	9	I	Y	i	y
xA	LF	SUB	*	:	J	Z	j	z
xB	VT	ESC	+	;	K	[	k	{
xC	FF	FS	,	<	L	\	l	¦
xD	CR	GS	-	=	M	]	m	}
xE	SO	RS	.	>	N	^	n	~
xF	SI	US	/	?	O	_	o	DEL

NUL carácter nulo = 00
SOH comienzo de cabecera de bloque
STX comienzo de texto
ETX fin de texto
EOT fin de transmisión
ENQ pregunta
ACK acuse de recepción positivo
BEL alarma sonora
BS borrado de caracteres
HT tabulación horizontal
LF línea siguiente (Line-Feed)
VT tabulación vertical
FF salto de página (Form-Feed)
CR retorno de carro
SO selección en salida
SI selección en entrada
SP espacio o blanco
DLE carácter de escape
DC1 servicio al periférico XON
DC2 mandato para el motor TAPE-ON
DC3 servicio al periférico XOFF
DC4 mandato para el motor TAPE-OFF
NAK acuse de recepción negativo
SYN carácter de sincronización
ETB fin de transmisión de bloque
CAN anulación (Cancel)
EM fin de mensaje
SUB carácter de sustitución
ESC carácter de escape
FS separador de fichero
GS separador de grupo
RS separador de registro
US separador de unidad
DEL carácter de rellenado

Anexo B

Relación de mandatos

Nombre del mandato	CP/M	MP/M	CP/M-86	MP/M-86
ABORT		X		X
ASM	X	X		
ASM86	X	X	X	X
ATTACH		X		X
CONSOLE		X		X
DDT	X	X		
DDT86			X	X
DIR	X	X	X	X
DSKRESET		X		X
DUMP	X	X	X	X
ED	X	X	X	X
ERA	X	X	X	X
ERAQ	X	X	X	X
GENCMD	X	X	X	X
GENHEX		X		
GENMOD		X		
GENSYS		X		X
LDCOPY			X	
LOAD	X	X		
MOVCPM	X			
MPMLDR	X			
MPMSTAT		X		X
PIP	X	X	X	X
PRINTER				X
PRLCOM		X		
REN	X	X	X	X
SAVE	X		X	
SCHED		X		X
SDIR				X
SET				X
SHOW				X
SPOOL		X		X
STAT	X	X	X	X
STOPSPLR		X		X
SUBMIT	X	X	X	X
SYSGEN	X			
TOD		X		X
TYPE	X	X	X	X
USER	X	X	X	X
XSUB	X	X	X	X

Anexo C

Relación de funciones de sistema

No. de función	Significado de la función CP/M	Parámetro de entrada	Parámetro de vuelta
C = 0	reinicialización del sistema	--	--
C = 1	lectura de un carácter en la consola	--	A=carácter
C = 2	escritura de un carácter en la consola	E=carácter	--
C = 3	lectura de un carácter en la lectora	--	A=carácter
C = 4	escritura de un carácter en la perforad.	E=carácter	--
C = 5	escritura de un carácter en la impresora	E=carácter	--
C = 6	lectura directa sobre la consola	E=FF(hexa)	A=carácter
C = 6	escritura directa sobre la consola	E=carácter	--
C = 7	lectura del octeto IOBYTE	--	A=IOBYTE
C = 8	inicialización del octeto IOBYTE	E=IOBYTE	--
C = 9	impresión de una cadena ASCII	DE=dirección	--
C = 10	lectura del buffer de consola	DE=dir.buf.	--
C = 11	lectura del "Status" de consola	--	A=0 ó FFH
C = 12	obtención del núm. de versión del sist.	--	HL=A=vers.
C = 13	reinicialización del disco de sistema	--	--
C = 14	selección de un disco	E=No.disco	--
C = 15	apertura de un fichero	DE=dir.FCB	A=cod.dir.
C = 16	cierre de un fichero	DE=dir.FCB	A=cod.dir.
C = 17	búsqueda del primer fichero ambiguo	DE=dir.FCB	A=cod.dir.
C = 18	búsqueda del fichero ambiguo siguiente	--	A=cod.dir.
C = 19	supresión de un fichero	DE=dir.FCB	A=cod.dir.
C = 20	lectura secuencial de un registro	DE=dir.FCB	A=cod.err.
C = 21	escritura secuencial de un registro	DE=dir.FCB	A=cod.err.
C = 22	creación de un fichero	DE=dir.FCB	A=cod.dir.
C = 23	cambio del nombre del fichero	DE=dir.FCB	A=cod.dir.
C = 24	lectura del vector de "Login"	--	HL=login
C = 25	lectura del número de disco en curso	--	HL=disco
C = 26	inicialización de direcc. de buffer DMA	DE=dir.DMA	--
C = 27	vuelta de la dirección de ubicación	--	HL=vector
C = 28	protección en escritura del disco	--	--
C = 29	lectura del vector de protección	--	HL=vector
C = 30	modificación de los atributos del fichero	DE=dir.FCB	A=cod.dir.
C = 31	dirección de los parámetros del disco	--	HL=dir.DPB
C = 32	lectura del número del usuario	E=FF (hexa)	A=número
C = 33	lectura directa sobre disco	DE=dir.FCB	A=cod.err.
C = 34	escritura directa sobre disco	DE=dir.FCB	A=cod.err.
C = 35	cálculo del tamaño del fichero	DE=dir.FCB	--
C = 36	posicionamiento sobre un registro	DE=dir.FCB	--

Número	Significado de la función MP/M	entrada	vuelta
C = 37	reinicialización de un "drive" de disco	DE=vector	A=0/FFH
C = 38	accesibilidad de un "drive" de disco	DE=vector	––
C = 39	liberación de un "drive" de disco	DE=vector	––
C = 40	escritura directa de un registro nulo	DE=dir.FCB	A=cod.ret.
C = 128	solicitud de asignación de memoria abs.	DE=dir.MD	A=0/FFH
C = 129	solic. asignac. de memoria relocatable	DE=dir.MD	A=0/FFH
C = 130	liberación de un segmento de memoria	DE=dir.MD	––
C = 131	investigación de un periférico (polling)	E=No.perif.	––
C = 132	puesta en espera de un suceso lógico	E=No.flag	A=0/FFH
C = 133	activación de un suceso lógico	E=No.flag	A=0/FFH
C = 134	creación de una cola (QCB)	DE=dir.QCB	––
C = 135	apertura de una cola (UQCB usuario)	DE=di.UQCB	A=0/FFH
C = 136	supresión de una cola (QCB)	DE=dir.QCB	A=0/FFH
C = 137	lectura incondicional de una cola	DE=di.UQCB	mens.leído
C = 138	lectura condicional de una cola	DE=di.UQCB	A=0/FFH
C = 139	escritura incondicional en una cola	DE=di.UQCB	––
C = 140	escritura condicional en una cola	DE=di.UQCB	A=0/FFH
C = 141	envío de una demora	DE=No.top.	––
C = 142	llamada del "dispatcher" para regulac.	––	––
C = 143	terminación normal de un proceso	DE=código	––
C = 144	creación de un proceso (Process descri.)	DE=dir.PD	PD creado
C = 145	modificación de la prioridad de un proc.	DE=prior.	––
C = 146	asignac. contr. un proceso desde consola	––	––
C = 147	liberación de un proceso de la consola	––	––
C = 148	asignac. y liberac. de un proc. a consola	––	––
C = 149	asignac. de la consola a un proceso	DE=dir.APB	A=0/FFH
C = 150	envío de un mandato al intérprete CLI	DE=CLICMD	––
C = 151	llamada de un proceso residente tipo RSP	DE=dir.CPB	HL=0/FFH
C = 152	análisis sintáctico del nombre del fichero	DE=dir.PFCB	––
C = 153	obtención del número de consola	––	A=número
C = 154	direc. zona de parámetros del sistema	––	HL=dir.
C = 155	obtención de fecha y hora	DE=dir.TOD	––
C = 156	direc. descriptor de proceso en curso	––	HL=dir.PD
C = 157	abortar un proceso	DE=dir.APB	A=código

Número	Significado de la función CP/NET	entrada	vuelta
C = 64	solicitud conex. con un sistema maestro	DE=mensaje	A=0/FFH
C = 65	desconexión de un usuario de la red	E=maestro	A=0/FFH
C = 66	envío de un mensaje a la red	DE=dir.men.	––
C = 67	recepción de un mensaje de la red	––	DE=dirbuf.
C = 68	estado de la red	E=ID sist.	A=estado
C = 69	dirección de la tabla de configuración	––	HL=direc.

Anexo D

Instrucciones máquina del 8080

MOV r1,r2	transfiere registro a registro	CPO dir	llamada si paridad impar
MOV M,r	transfiere registro a memoria	RET	vuelta de subrutina
MOV r,M	transfiere memoria a registro	RC	vuelta si hay acarreo
HLT	alto de la CPU	RNC	vuelta si no hay acarreo
MVI r,v	carga inmediata del registro	RZ	vuelta si cero
MVI M,v	carga inmediata de memoria	RNZ	vuelta si no nulo
INR r	incremento de registro	RP	vuelta si positivo
DCR r	decremento de registro	RM	vuelta si negativo
INR M	incremento de memoria	RPE	vuelta si paridad par
DCR M	decremento de memoria	RPO	vuelta si paridad impar
ADD r	sumar registro a A	RST	vuelta bajo interrupción
ADC r	sumar en A con acarreo	IN	lectura de periférico
SUB r	restar registro de A	OUT	escritura en un periférico
SBB r	restar con acarreo	LXI B,v	carga inmediata de BC
ANA r	intersección de registro y A	LXI D,v	carga de DE
XRA r	OR exclusivo del registro y A	LXI H,v	carga de HL
ORA r	reunión del registro con A	LXI SP,v	carga de SP
CMP r	comparación de registro con A	PUSH B	puesta en pila de BC
ADD M	sumar en memoria con A	PUSH D	puesta en pila de DE
ADC M	sumar en memoria con acarreo	PUSH H	puesta en pila de HL
SUB M	resta en memoria de A	PUSH PSW	puesta en pila de A y PSW
SBB M	resta con acarreo	POP B	sacar de pila de BC
ANA M	AND de memoria con A	POP D	sacar de pila de DE
XRA M	OR exclusivo de memoria con A	POP H	sacar de pila de HL
ORA M	OR de memoria con A	POP PSW	sacar de pila de A y PSW
CPM M	comparación de memoria con A	STA dir	almacenamiento directo de A
ADI v	suma inmediata con A	LDA dir	carga directa de A
ACI v	suma inmediata de A con CY	XCHG	intercambio de DE y HL
SUI v	resta inmediata con A	XTHL	intercambio de SP y HL
SBI v	resta con acarreo	SPHL	forzar HL en SP
ANI v	AND inmediato con A	PCHL	forzar el contador ordinal
XRI v	OR exclusivo inmediato con A	DAD B	suma de BC en HL
ORI v	OR inmediato con A	DAD D	suma de DE en HL
CPI v	comparación inmediata con A	DAD H	suma de HL en HL

DAD SP	suma de SP en HL	CNZ dir	llamada si resultado no nulo
STAX B	almacenamiento indirecto de A	CP dir	llamada si resultado positivo
STAX D	almacenamiento indirecto de A	CM dir	llamada si resultado negativo
LDAX B	carga indirecta de A	CPE dir	llamada si paridad par
RLC n	rotación a izquierda de A	LDAX D	carga indirecta de A
RRC n	rotación de derecha de A	INX B	incrementar BC
RAL n	rotación izquierda a través de CY	INX D	incrementar DE
RAR n	rotación derecha a través de CY	INX H	incrementar HL
JMP dir	bifurcación incondicional	INX SP	incrementar SP
JC dir	bifurcación si hay acarreo	DCX B	decrementar BC
JNC dir	bifurcación si no hay acarreo	DCX D	decrementar DE
JZ dir	bifurcación si resultado nulo	DCX H	decrementar HL
JNZ dir	bifurcación si no nulo	DCX SP	decrementar SP
JP dir	bifurcación si positivo	CMA	complementar A
JM dir	bifurcación si negativo	STC	forzar acarreo
JPE dir	bifurcación si paridad par	CMC	complementar acarreo
JPO dir	bifurcación si paridad impar	DAA	suma decimal
CALL dir	llamada a la subrutina	SHLD dir	almacenamiento directo de HL
CC dir	llamada si hay acarreo	LHLD dir	carga directa de HL
CNC dir	llamada si no hay acarreo	EI	autorizar interrupciones
CZ dir	llamada si resultado nulo	DI	inhibe las interrupciones
		NOP	no operación

Anexo E

Principales tipos de ficheros

COM:	fichero ejecutable en la zona TPA (CP/M y MP/M)
ASM:	fichero fuente en ensamblador 8080 (CP/M y MP/M)
HEX:	fichero objeto en formato Intel 8080 en hexadecimal
PRN:	fichero imagen de impresora
BAK:	fichero imagen de la última versión tratado por ED
$$$:	fichero temporal de trabajo
BAS:	fichero cargable por el intérprete MBASIC
COB:	fichero fuente escrito en lenguaje COBOL
FOR:	fichero fuente escrito en lenguaje FORTRAN
PAS:	fichero fuente escrito en lenguaje PASCAL
PCO:	fichero intermedio en código P de PASCAL
INT:	fichero en código intermedio
ASC:	fichero fuente escrito en lenguaje BASIC
DAT:	fichero de datos
DOC:	fichero de documentos
LIB:	fichero biblioteca
MAC:	fichero fuente para el macro-ensamblador MAC-80
SUB:	fichero catálogo de mandatos a submitir
PRL:	fichero relocatable ejecutable en un segmento de memoria bajo MP/M
REL:	fichero objeto en código intermedio relocatable por LINK–80
TXT:	fichero de texto
RSP:	fichero programa de sistema que puede ser residente en memoria (MP/M)
SPR:	fichero módulo de sistema MP/M
SYS:	fichero imagen del disco de sistema (MP/M, CP/M-86, MP/M-86)
A86:	fichero fuente en ensamblador 8086
H86:	fichero objeto en formato Intel 8086 en hexadecimal
LST:	fichero imagen de impresión de un ensamblaje bajo ASM86
CMD:	fichero ejecutable bajo CP/M-86 y MP/M-86

Anexo F

Principales productos bajo CP/M

Lenguaje	Nom. Producto.	Descripción del producto	Creador
MACRO	MAC	macro-ensamblador 8080	Digital-Research
	MACRO-80	macro-ensamblador 8080/Z80	Microsoft
BASIC	MBASIC	intérprete	Microsoft
	CBASIC	intérprete	Compiler Systems
	KBASIC	intérprete	Eidos
	XY BASIC	intérprete	M. Williams Co.
	BASCOM	compilador	Microsoft
	SBASIC	compilador estructurado	Topaz Programming
PASCAL	PASCAL/M	intérprete (código P)	Sorcim
	PASCAL/MT+	compilador (norma ISO)	MT Microsystems
	PASCAL/Z	compilador para Z80	Ithaca Intersystems
	JRT PASCAL	intérprete tiempo real	JRT
FORTRAN	FORTRAN-80	compilador (ANSI-66)	Microsoft
	SSS FORTRAN	compilador	Supersoft
COBOL	COBOL-80	compilador (ANSI-74)	Microsoft
	CIS-COBOL	intérprete (ANSI-74)	Micro-Focus
	NEVADA-COBOL	compilador (ANSI-74)	Ellis
C	C COMPILER	compilador	Whitesmiths
	BDS-C	compilador	BDS
	C	compilador	Supersoft
	Tiny C	intérprete	Tiny C Associated
	C86	compilador para 8086	Computer Innovation
PL1	PL/1-80	compilador sub-conjunto	Digital Research
PL/M	PL/M	compilador	Digital Research
LISP	mu-LISP	intérprete	Microsoft
	LISP	intérprete	Supersoft
APL	uAPL	intérprete	Softronics
ALGOL	ALGOL-60	compilador	Research
ADA	ADA	compilador sub-conjunto	Supersoft
FORTH	FORTH	intérprete	Supersoft
RATFOR	RATFOR	traductor en FORTRAN	Supersoft

Herramientas de desarrollo

Nombre del producto	Descripción del producto	Fabricante
SID, ZSID	"depuradores" simbólicos 8080 y Z80	Digital Research
EDIT-80	editor de texto modo línea	Microsoft
VEDIT	editor de texto modo página	CompuView Products
IBMCPM	conversión de disketes IBM-3740	Lifeboat Ass.
TRANS86	traductor de binarios 8080/8086	Sorcim
DISTEL, DISILOG	desensambladores 8080 y Z80	Lifeboat Ass.

Tratamiento de textos

Nombre del producto	Descripción del producto	Fabricante
WORDSTAR	editor de textos y documentos	Micropro
TEX	editor de textos modo página	Digital Research
TEXWRITER III	editor de textos y documentos	Organic Software
WORDMASTER	editor de textos modo página	Micropro
SPELLBINDER	editor de documentos, fotocomposición	Lexisoft
SPELLSTAR	verificador de documentos	Micropro
SPELLGUARD	verificador de documentos	Innovative Software

Paquetes de gestión

Nombre del producto	Descripción del producto	Fabricante
DATASTAR	captura y consulta de ficheros	Micropo
CALCSTAR	gestión financiera y previsiones	Micropo
SUPERCALC	gestión financiera y previsiones	Sorcim
DBASE II	sistema de base de datos	Ashton-Tate
MAIL-MERGE	sistema de "correo electrónico"	Microsoft

Anexo G

Algunos micro-ordenadores

CP/M	CP/M y MP/M	CP/M-86 y MP/M-86
Apple II + Softcard Z80	Addx systèmes SM1/SM2	IF86 Infor Française
Cromenco System 3	Altos ACS8000/TKL8000	Altos ACS8600
Datapoint 1550/2150	Dynabyte DB 8/4	Micromachine 4000
Dec Rainbow 100	Intel MDS/800	NSC 6604/6608
Heath H8/H89	Micromation	Sirius 1
Hewlett-Packard HP 125	Onyx C8001	GPS Système 101
IF-800	Quasar	IBM PC
ITT 3030	System 2800	MICRAL 9050
JB 3000 Panasonic	Zobex	GOUPIL 3
Logabax LX500	Z-Plus	SILZ 16
Micropolis mod I/II		NCR Décision V
Mostek MDX		
Ohio Scientific C3		
Osborne 1		
Northstar Horizon		
Pertec PCC-1000/2000		
Radio Shack TRS80 I/II		
Sanco 2000/7000		
Stratos		
TKL-10, TKL-20		
Xerox 820 microcomputer		
Zenith Z89		

WATT y MANGADA.– BASIC para niños. 5ª edición.
WATT y MANGADA.– BASIC avanzado para niños. 3ª edición.
WATT y MANGADA.– BASIC para niños con el microordenador DRAGON.

Aplicaciones e Informática Profesional

ANGULO.– Curso de Robótica. 2ª edición.
ANGULO.– Robótica práctica. Teoría y aplicaciones.
ANGULO.– Prácticas de Microelectrónica y microinformática. 2ª edición.
ASPINALL.– El microprocesador y sus aplicaciones.
BANKS.– Microordenadores. Cómo funcionan. Para qué sirven.
BELLIDO.– Amaestra tu DRAGON. Curso de programación en BASIC para el microordenador DRAGON.
BELLIDO.– ZX81. Curso de programación en BASIC. 3ª edición.
BELLIDO.– Cómo programar su Spectrum y Timex 2068. 7ª edición.
BELLIDO.– Cómo usar los colores y los gráficos en el Spectrum. 3ª edición. (Libro y casete).
BELLIDO.– KIT de gráficos para Spectrum.
BELLIDO.– Enciclopedia del Spectrum. Tomo 1.
BELLIDO.– Spectrum. Iniciación al Código Máquina.
ELLERSHAW y SCHOFIELD.– Las primeras 15 horas con el Spectrum.
ERSKINE.– Los mejores programas para el ZX Spectrum.
ESCUDERO.– (Centro de Investigación UAM-IBM). Reconocimiento de patrones. Fundamentos teóricos, algorítmos y aplicaciones de la moderna técnica denominada "Pattern Recognition".
FERRER.– Programas en BASIC.
GAUTHIER y PONTO.– Diseño de programas para sistemas. 4ª edición.
HARTMAN, MATTHES y PROEME.– Manual de los sistemas de información. 2 tomos. 7ª edición.
LEPAPE.– Programación del Z80 con ensamblador.
LUCAS, JR.– Sistemas de información. Análisis. Diseño. Puesta a punto.
MARTINEZ VELARDE.– El libro de Código Máquina del Spectrum. 2ª edición.
MONTEIL.– Cómo programar su Commodore 64. 2 tomos. Tomo 1: BASIC. Gráficos. Sonidos. 4ª edición. Tomo 2: Lenguaje máquina. Entradas-salidas y periféricos.
PANNELL, JACKSON y LUCAS.– El microordenador en la pequeña Empresa.
PLOUIN.– IBM-PC. Características. Programación. Manejo.
QUANEAUX.– Tratamiento de textos en BASIC.
WILLIAMS.– Programación paso a paso con el Spectrum.

CP/M. Guía de utilización

La obra está enfocada hacia aquellas personas que deseen comprender y practicar el CP/M, MP/M y sus ramificaciones.

CP/M se ha convertido en un verdadero estándar para los micro-ordenadores.

El éxito del CP/M radica en sí mismo y en la gran variedad de programas que a su amparo han sido desarrollados.

En esta obra, además de la descripción completa del funcionamiento y mandos del CP/M y MP/M, hallará una guía de las ramificaciones y extensiones actuales que han surgido del CP/M, de MP/M y de los micro-procesadores de 16 bits: CP/M-86, MP/M-86, CP/NET, MP/NET, etc.

Como complemento, se dedica un capítulo especial a los productos desarrollados en torno al CP/M y MP/M: Lenguajes de programación, tratamiento de texto y programas de gestión.



Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1430-6

PARANINFO S.A.

CP/M Guía de utilización

Philippe DAX

AMSTRAD CPC

OCR

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA



https://acpc.me/